I Semester	ADVANCED DATA WAREHOUSING AND DATA MINING LAB	L	T	P	C
		0	1	2	2

Pre-requisites: Data Base Management Systems, Python Programming

COURSE OBJECTIVES: The main objective of the course is to

- Inculcate Conceptual, Logical, and Physical design of Data Warehouses OLAP applications and OLAP deployment
- Design a data warehouse or data mart to present information needed by management in a form that is usable
- Emphasize hands-on experience working with all real data sets.
- Test real data sets using popular data mining tools such as WEKA, Python Libraries
- Develop ability to design various algorithms based on data mining tools.

Software Requirements: WEKA Tool/Python/R-Tool/Rapid Tool/Oracle Data mining

UNIT	CONTENTS	Contact Hours
Experiment– 1	Creation of a Data Warehouse. ➢ Build Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration Tool, Pentaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects,etc.,) ➢ Design multi-dimensional data models namely Star, Snowflake and Fact Constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, manufacturing, Automobiles, sales etc). ➢ Write ETL scripts and implement using data warehouse tools. ➢ Perform Various OLAP operations such slice, dice, roll up, drill up and pivot	10Hrs
Experiment– 2	Explore machine learning tool “WEKA” ➢ Explore WEKA Data Mining/Machine Learning Toolkit. ➢ Downloading and/or installation of WEKA data mining toolkit. ➢ Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface, Experimenter, command-line interface. ➢ Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, Classify panel, Cluster panel, Associate panel and Visualize panel) ➢ Study the arff file format Explore the available data sets in WEKA. Load a data set (ex. Weather dataset, Iris dataset, etc.) ➢ Load each dataset and observe the following:	12Hrs

	1. List the attribute names and they types 2. Number of records in each dataset 3. Identify the class attribute (if any) 4. Plot Histogram 5. Determine the number of records for each class. 6. Visualize the data in various dimensions	
Experiment– 3	Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets ➢ Explore various options available in Weka for preprocessing data and apply Unsupervised filters like Discretization, Resample filter, etc. on each dataset ➢ Load weather. nominal, Iris, Glass datasets into Weka and run Apriori Algorithm with different support and confidence values. ➢ Study the rules generated. Apply different discretization filters on numerical attributes and run the Apriori association rule algorithm. Study the rules generated. ➢ Derive interesting insights and observe the effect of discretization in the rule generation process.	12Hrs
Experiment– 4	Demonstrate performing classification on data sets Weka/R ➢ Load each dataset and run 1d3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic. ➢ Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix. ➢ Load each dataset into Weka/R and perform Naïve-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained. ➢ Plot RoC Curves ➢ Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.	
Experiment– 5	Demonstrate performing clustering of data sets ➢ Load each dataset into Weka/R and run simple k-means clustering algorithm with different values of k (number of desired clusters). ➢ Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights. ➢ Explore other clustering techniques available in Weka/R. Explore visualization features of Weka/R to visualize the clusters. Derive interesting insights and explain	12Hrs

Experiment– 6	Demonstrate knowledge flow application on data sets into Weka/R ➢ Develop a knowledge flow layout for finding strong association rules by using Apriori, FP Growth algorithms ➢ Set up the knowledge flow to load an ARFF (batch mode) and perform a cross validation using J48 algorithm Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree	12Hrs
Experiment– 7	Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observations	
Experiment– 8	Write a java program to prepare a simulated data set with unique instances	
Experiment– 9	Write a Python program to generate frequent item sets / association rules using Apriori algorithm	
Experiment– 10	Write a program to calculate chi-square value using Python/R. Report your observation.	
Experiment– 11	Implement a Java/R program to perform Apriori algorithm	
Experiment– 12	Write a R program to cluster your choice of data using simple k-means algorithm using JDK	
Experiment– 13	Write a program of cluster analysis using simple k-means algorithm Python/R programming language	
Experiment– 14	Write a program to compute/display dissimilarity matrix (for your own dataset containing at least four instances with two attributes) using Python	
Experiment– 15	Visualize the datasets using matplotlib in python/R.(Histogram, Box plot, Bar chart, Pie chart etc.,)	
	Total	58Hrs

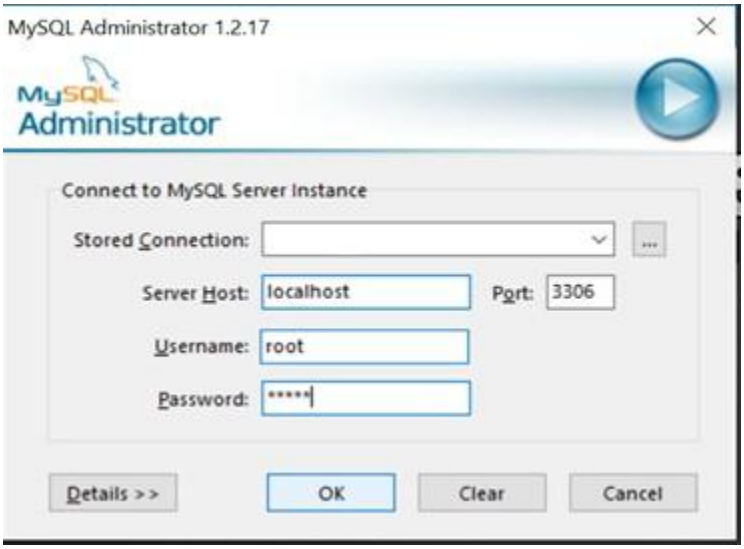
Data Warehousing
Experiments:

1. Build Data Warehouse and Explore WEKA

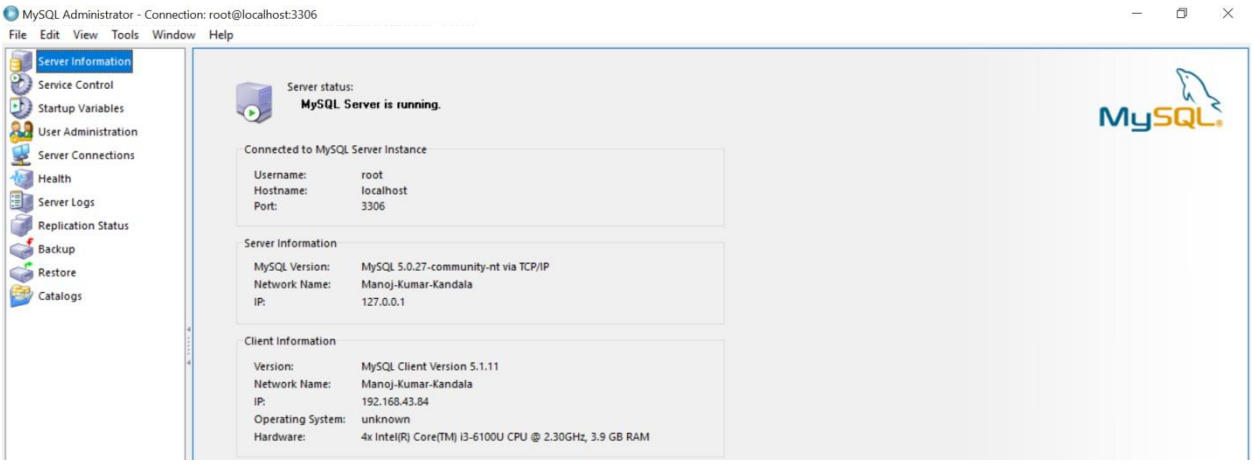
A. Build a Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration tool, Pentaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects, etc.).

(i). Identify source tables and populate sample data

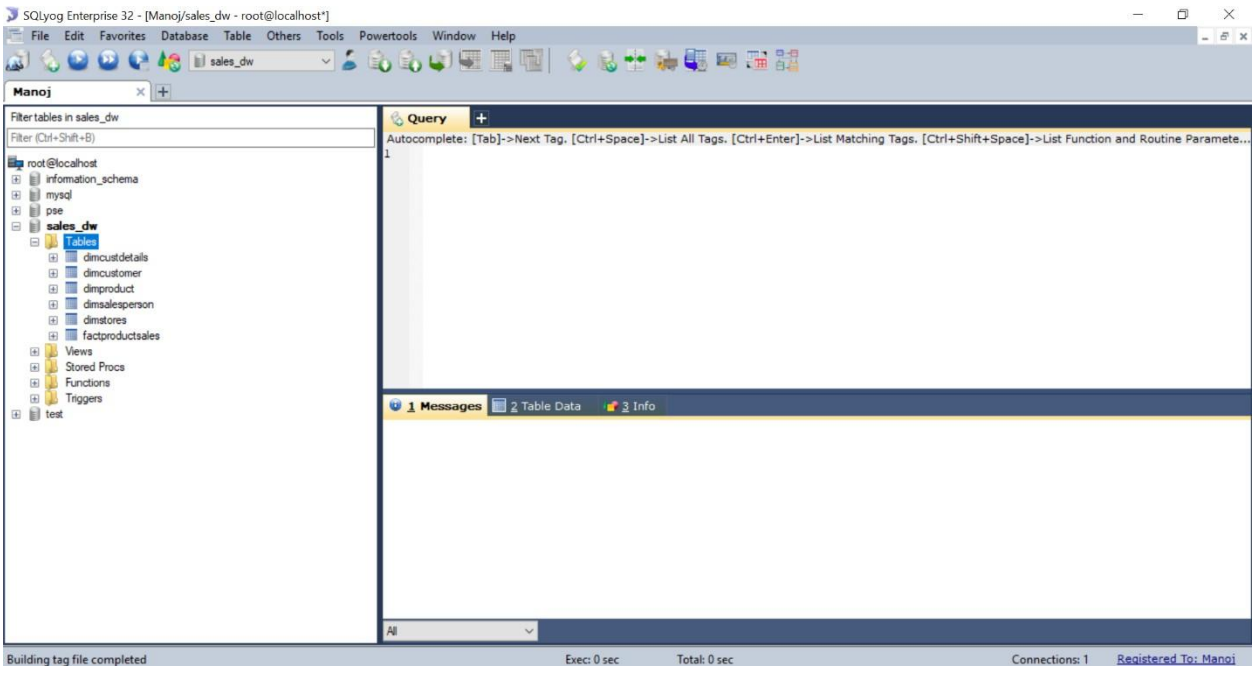
In this task, we are going to use **MySQL administrator, SQLyog Enterprise tools** for building & identifying tables in database & also for populating (filling) the sample data in those tables of a database.A data warehouse is constructed by integrating data from multiple heterogeneous sources. It supports analytical reporting, structured and/or ad hoc queries and decision making. We are building a data warehouse by integrating all the tables in database & analyzing those data. In the below figure we represented MySQL Administrator connection establishment.



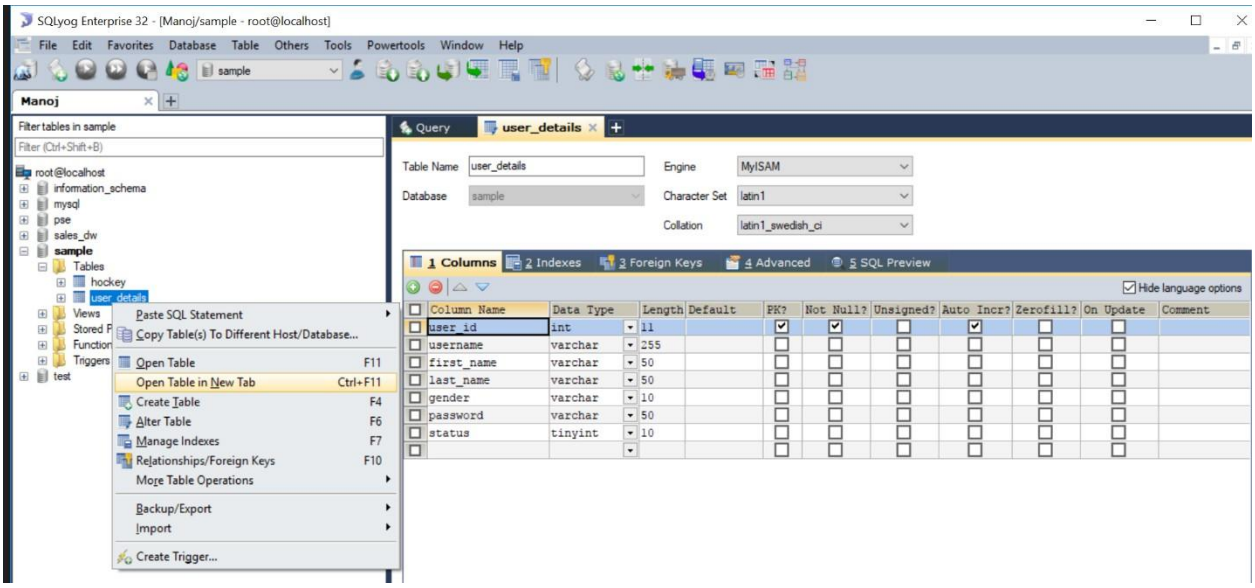
After successful login, it will open new window as shown below.

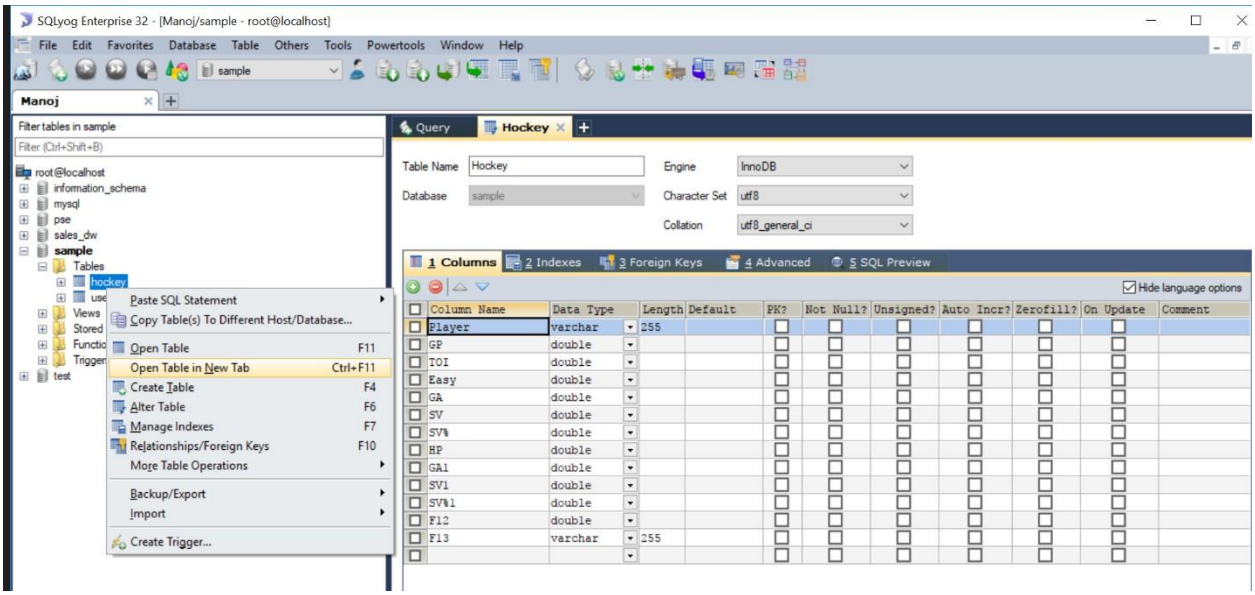


There are different options available in MySQL administrator. Another tool SQLyog Enterprise, we are using for building & identifying tables in a database after successful connection establishment through MySQL Administrator. Below we can see the window of SQLyog Enterprise.



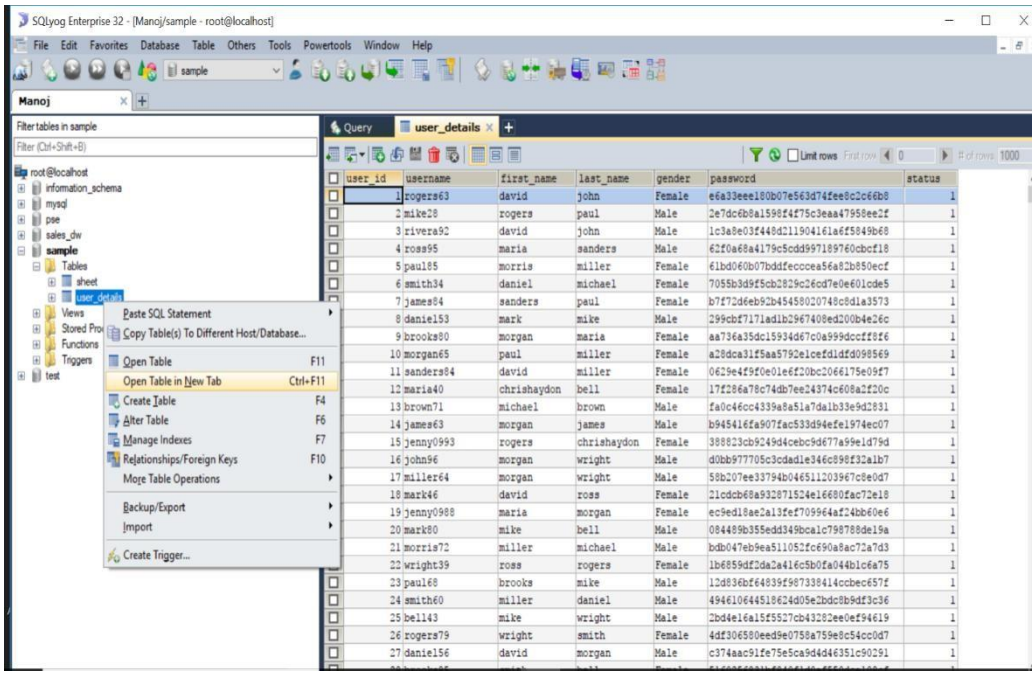
On left-side navigation, we can see different databases & it‘s related tables. Now we are going to build tables & populate table‘s data in database through SQL queries. These tables in database can be used further for building data warehouse.

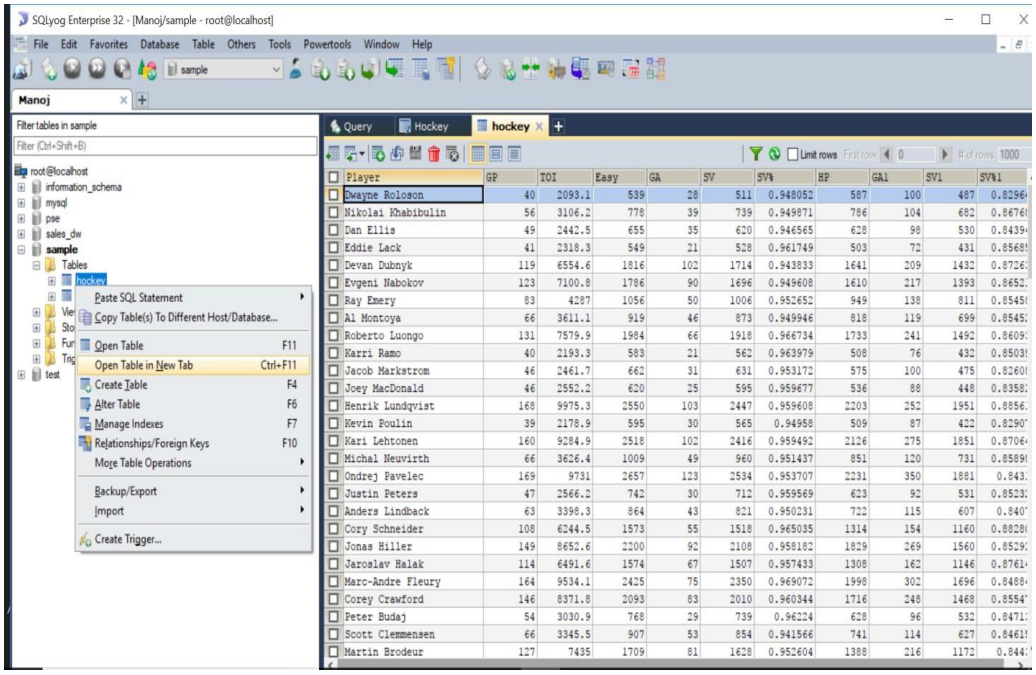




In the above two windows, we created a database named **"sample"**& in that database we created two tables named as **"user_details"**& **"hockey"**through SQL queries.

Now, we are going to populate (filling) sample data through SQL queries in those two created tables as represented in below windows.


SQLyog Enterprise 32 - [Manoj/sample - root@localhost]
File Edit Favorites Database Table Others Tools Powertools Window Help
Manoj
Filter tables in sample
root@localhost
information_schema
mysql
pse
sales_dw
sample
Tables
sheet
user_details
Views
Stored Pro
Functions
Triggers
test
Query
user_details
Paste SQL Statement
Copy Table(s) To Different Host/Database...
Open Table F11
Open Table in New Tab Ctrl+F11
Create Table F4
Alter Table F6
Manage Indexes F7
Relationships/Foreign Keys F10
More Table Operations
Backup/Export
Import
Create Trigger...
user_id username first_name last_name gender password status
1 rogers63 david john Female e6a33eee180b07e563d74fee8c2c66b8 1
2 mike28 rogers paul Male 2e7dc6b8a1598f4f75c3eaa47958ee2f 1
3 rivera92 david john Male 1c3a8e03f448d21190416la6f5849b68 1
4 ross95 maria sanders Male 62f0a68a4179c5cdd997189760cbcf18 1
5 paul85 morris miller Female 61bd060b07bddfecccea56a82b850ecf 1
6 smith34 daniel michael Female 7055b3d9f5cb2829c26cd7e0e601cde5 1
7 james84 sanders paul Female b7f72d6eb92b45458020748c8d1a3573 1
8 daniel53 mark mike Male 299cbf717ad1b2967408ed200b4e26c 1
9 brooks80 morgan maria Female aa736a35dc15934d67c0a999dccff8f6 1
10 morgan65 paul miller Female a28dca31f5aa5792e1cefd1dfd098569 1
11 sanders84 david miller Female 0629e4f9f0e01e6f20bc2066175e09f7 1
12 maria40 chrishaydon bell Female 17f286a78c74db7ee24374c608a2f20c 1
13 brown71 michael brown Male fa0c46cc4339a8a51a7da1b33e9d2831 1
14 james63 morgan james Male b945416fa907fac533d94efe1974ec07 1
15 jenny0993 rogers chrishaydon Female 388823cb9249d4cebc9d677a99e1d79d 1
16 john96 morgan wright Male d0bb977705c3cdad1e346c898f32a1b7 1
17 miller64 morgan wright Male 58b207ee33794b046511203967c8e0d7 1
18 mark46 david ross Female 21cdcb68a932871524e16680fac72e18 1
19 jenny0988 maria morgan Female ec9ed18ae2a13fef70996 4af24bb60e6 1
20 mark80 mike bell Male 084489b355edd349bca1c798788de19a 1
21 morris72 miller michael Male bdb047eb9ea511052fc690a8ac72a7d3 1
22 wright39 ross rogers Female 1b6859df2da2a416c5b0fa044b1c6a75 1
23 paul68 brooks mike Male 12d836bf64839f987338414ccbec657f 1
24 smith60 miller daniel Male 494610644518624d05e2bdc8b9df3c36 1
25 bell43 mike wright Male 2bd4e16a15f5527cb43282ee0ef94619 1
26 rogers79 wright smith Female 4df306580eed9e0758a759e8c54cc0d7 1
27 daniel56 david morgan Male c374aac91fe75e5ca9d4d46351c90291 1



Through MySQL administrator & SQLyog, we can import databases from other sources (.XLS, .CSV, .sql) & also we can export our databases as backup for further processing. We can connect MySQL to other applications for data analysis & reporting.

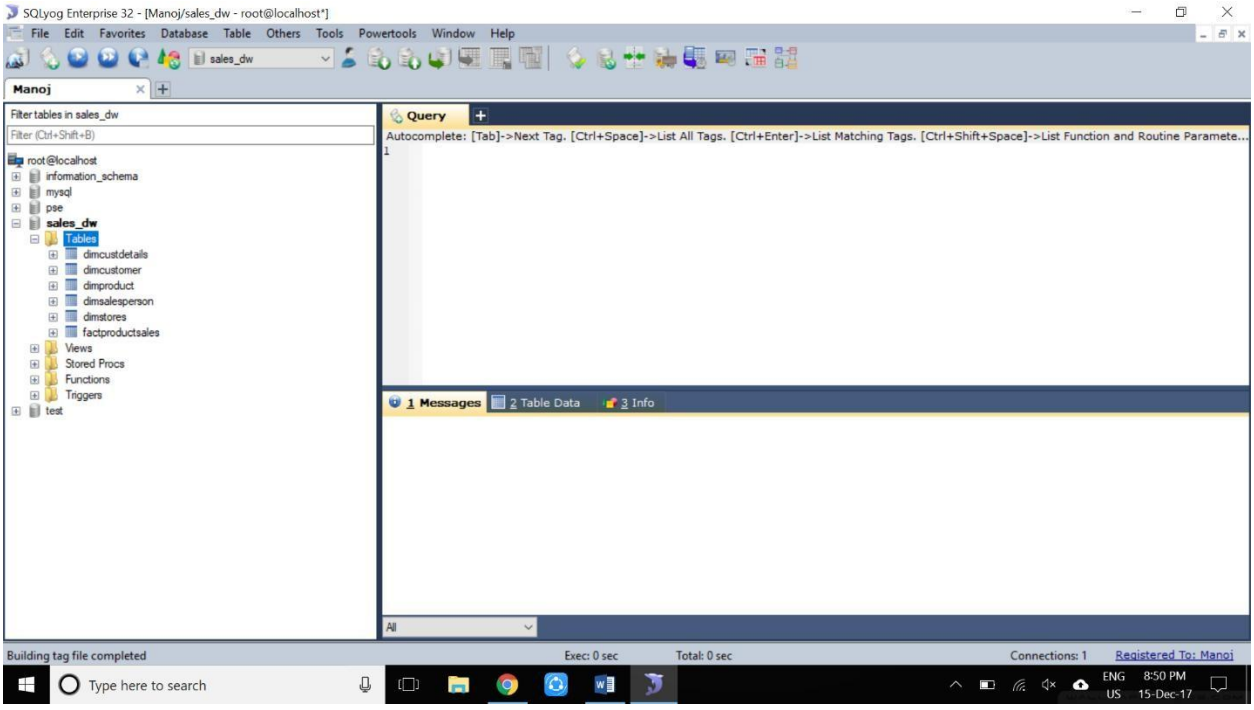
(ii). Design multi-dimensional data models namely Star, snowflake and Fact constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, Manufacturing, Automobile, etc.).

Multi-Dimensional model was developed for implementing data warehouses & it provides both a mechanism to store data and a way for business analysis. The primary components of dimensional model are dimensions & facts. There are different of types of multi-dimensional data models. They are:

1. Star Schema Model
2. Snow Flake Schema Model
3. Fact Constellation Model.

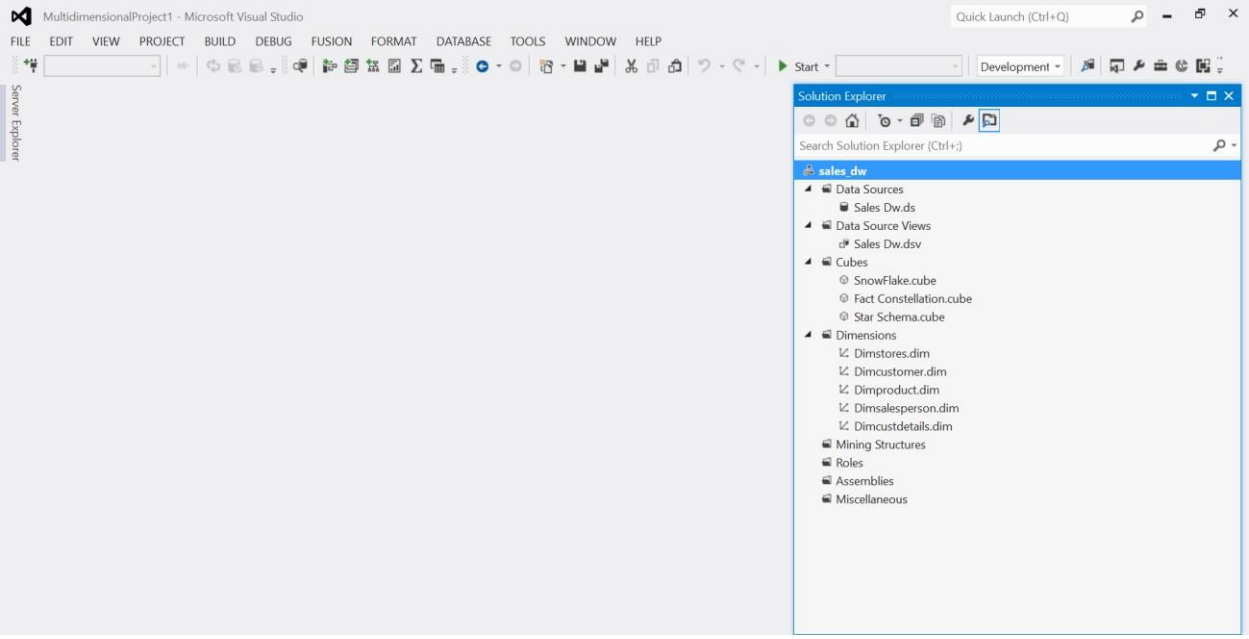
Now, we are going to design these multi-dimensional models for the Marketing enterprise.

First, we need to built the tables in a database through SQLyog as shown below.



In the above window, left side navigation bar consists of a database named as ―sales_dw‖ in which there are six different tables (dimcustdetails, dimcustomer, dimproduct, dimsalesperson, dimstores, factproductsales) has been created.

After creating tables in database, here we are going to use a tool called as **"Microsoft Visual Studio 2012 for Business Intelligence"** for building multi-dimensional models.



In the above window, we are seeing Microsoft Visual Studio before creating a project In which right side navigation bar contains different options like Data Sources, Data Source Views, Cubes, Dimensions etc.

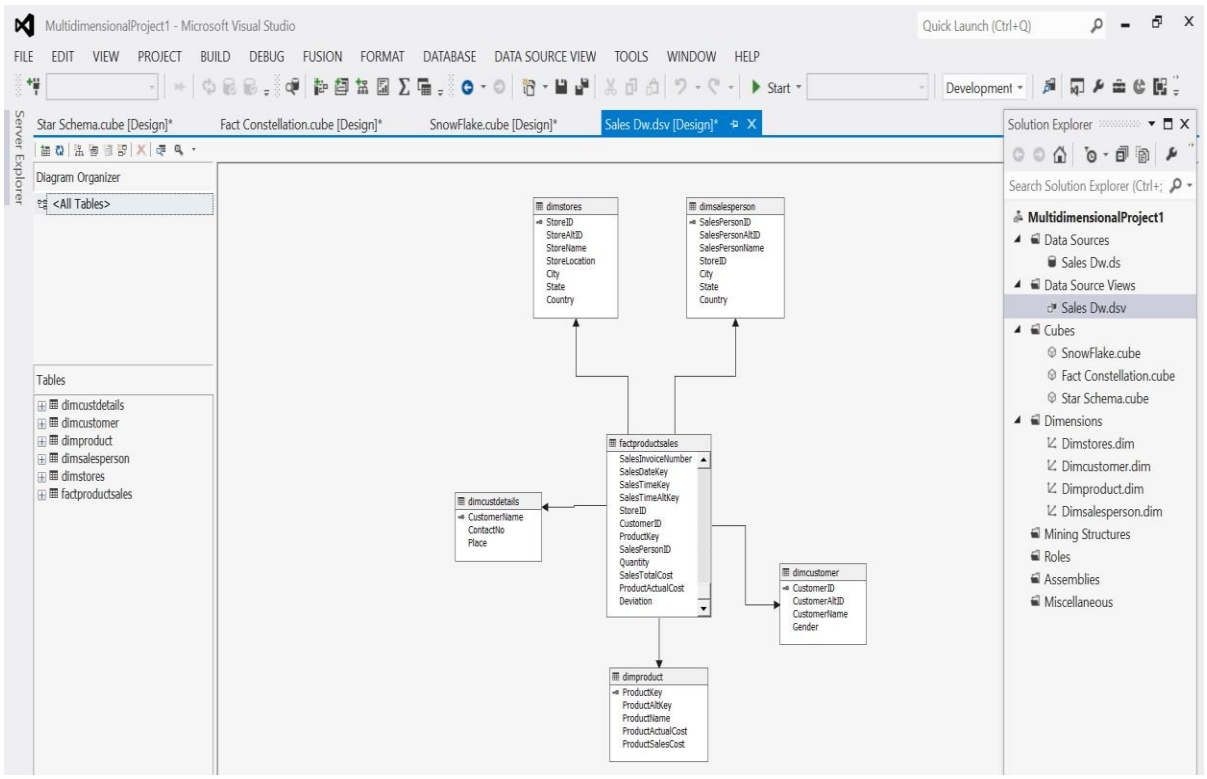
Through Data Sources, we can connect to our MySQL database named as **"sales_dw".** Then, automatically all the tables in that database will be retrieved to this tool for creating multi-dimensional models.

By data source views & cubes, we can see our retrieved tables in multi-dimensional models. We need to add dimensions also through dimensions option. In general, Multi-dimensional models consists of dimension tables & fact tables.

Star Schema Model:

A Star schema model is a join between a fact table and a no. of dimension tables. Each dimensional table are joined to the fact table using primary key to foreign key join but dimensional tables are not joined to each other. It is the simplest style of dataware house schema.

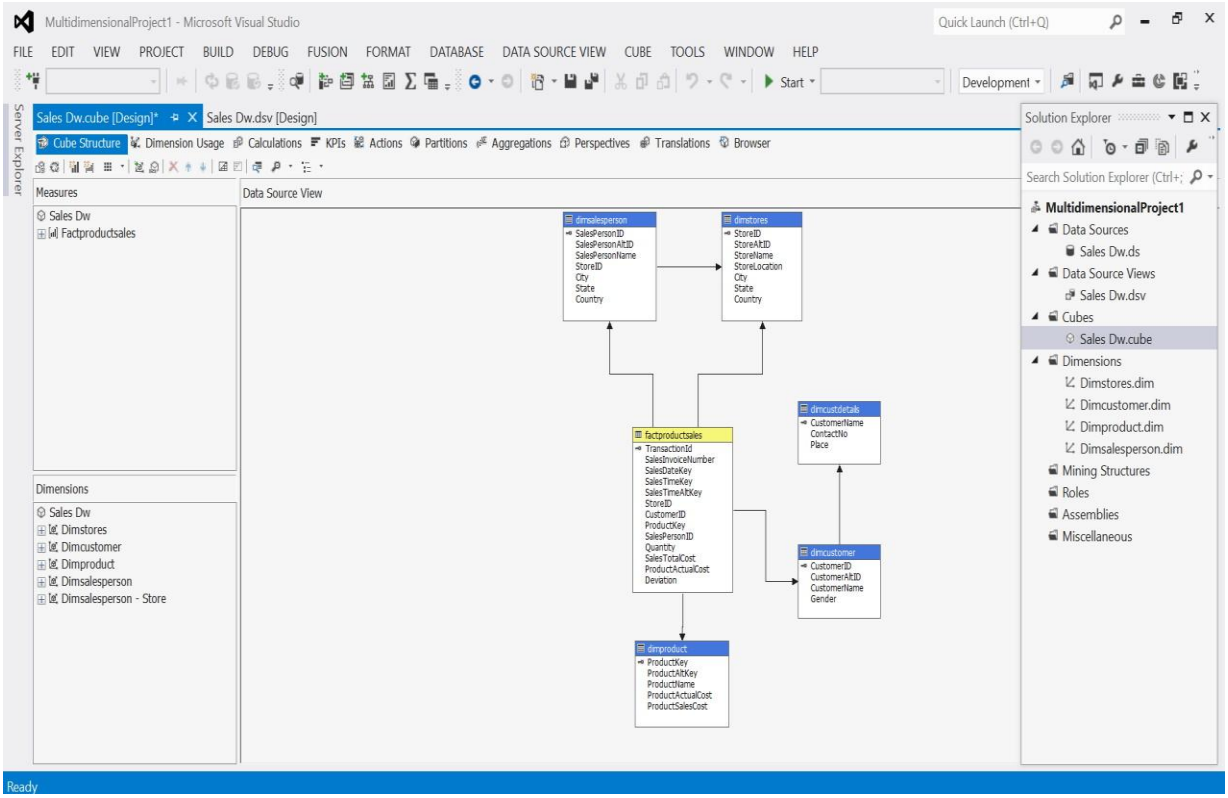
Star schema is a entity relationship diagram of this schema resembles a star with point radiating from central table as we seen in the below implemented window in visual studio.



Snow Flake Schema:

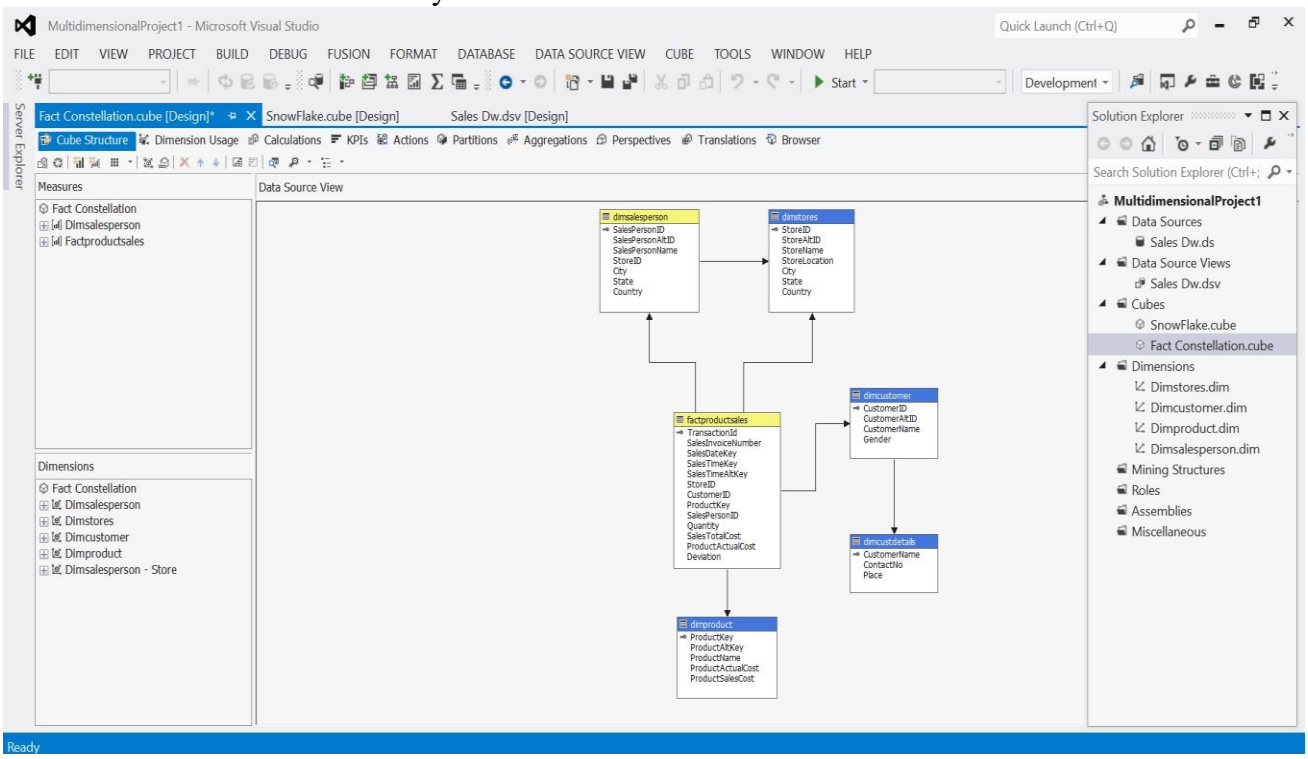
It is slightly different from star schema in which dimensional tables from a star schema are organized into a hierarchy by normalizing them.

Snow flake schema is represented by centralized fact table which are connected to multiple dimension tables. Snow flake effects only dimension tables not fact tables. we developed a snowflake schema for sales_dw database by visual studio tool as shown below.



Fact Constellation Schema:

Fact Constellation is a set of fact tables that share some dimension tables. In this schema there are two or more fact tables. We developed fact constellation in visual studio as shown below. Fact tables are labelled in yellow color.



2. Write ETL scripts and implement using data warehouse tools

ETL (Extract-Transform-Load):

ETL comes from Data Warehousing and stands for Extract-Transform-Load. ETL covers a process of how the data are loaded from the source system to the data warehouse. Currently, the ETL encompasses a cleaning step as a separate step. The sequence is then Extract-Clean-Transform-Load. Let us briefly describe each step of the ETL process.

Process

Extract:

The Extract step covers the data extraction from the source system and makes it accessible for further processing. The main objective of the extract step is to retrieve all the required data from the source system with as little resources as possible. The extract step should be designed in a way that it does not negatively affect the source system in terms or performance, response time or any kind of locking.

There are several ways to perform the extract:

- Update notification - if the source system is able to provide a notification that a record has been changed and describe the change, this is the easiest way to get the data.
- Incremental extract - some systems may not be able to provide notification that an update has occurred, but they are able to identify which records have been modified and provide an extract of such records. During further ETL steps, the system needs to identify changes and propagate it down. Note, that by using daily extract, we may not be able to handle deleted records properly.
- Full extract - some systems are not able to identify which data has been changed at all, so a full extract is the only way one can get the data out of the system. The full extract requires keeping a copy of the last extract in the same format in order to be able to identify changes. Full extract handles deletions as well.

When using Incremental or Full extracts, the extract frequency is extremely important. Particularly for full extracts; the data volumes can be in tens of gigabytes.

Clean:

The cleaning step is one of the most important as it ensures the quality of the data in the data warehouse. Cleaning should perform basic data unification rules, such as:

- Making identifiers unique (sex categories Male/Female/Unknown, M/F/null, Man/Woman/Not Available are translated to standard Male/Female/Unknown)
- Convert null values into standardized Not Available/Not Provided value
- Convert phone numbers, ZIP codes to a standardized form
- Validate address fields, convert them into proper naming, e.g. Street/St/St./Str./Str
- Validate address fields against each other (State/Country, City/State, City/ZIP code, City/Street).

Transform:

The transform step applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension (i.e. conformed dimension) using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

Load:

During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

Managing ETL Process:

The ETL process seems quite straight forward. As with every application, there is a possibility that the ETL process fails. This can be caused by missing extracts from one of the systems, missing values in one of the reference tables, or simply a connection or power outage. Therefore, it is necessary to design the ETL process keeping fail-recovery in mind.

Staging:

It should be possible to restart, at least, some of the phases independently from the others. For example, if the transformation step fails, it should not be necessary to restart the Extract step. We can ensure this by implementing proper staging. Staging means that the data is simply dumped to the location (called the Staging Area) so that it can then be read by the next processing phase. The staging area is also used during ETL process to store intermediate results of processing. This is ok for the ETL process which uses for this purpose. However, tThe staging area should is be accessed by the load ETL process only. It should never be available to anyone else; particularly not to end users as it is not intended for data presentation to the end-user.may contain incomplete or in-the-middle-of-the-processing data.

ETL Tool Implementation:

When you are about to use an ETL tool, there is a fundamental decision to be made: will the company build its own data transformation tool or will it use an existing tool?

Building your own data transformation tool (usually a set of shell scripts) is the preferred approach for a small number of data sources which reside in storage of the same type. The reason for that is the effort to implement the necessary transformation is little due to similar data structure and common system architecture. Also, this approach saves licensing cost and there is no need to train the staff in a new tool. This approach, however, is dangerous from the TOC point of view. If the transformations become more sophisticated during the time or there is a need to integrate other systems, the complexity of such an ETL system grows but the manageability drops significantly. Similarly, the implementation of your own tool often resembles re-inventing the wheel.

There are many ready-to-use ETL tools on the market. The main benefit of using off-the-shelf ETL tools is the fact that they are optimized for the ETL process by providing connectors to common data sources like databases, flat files, mainframe systems, xml, etc. They provide a means to implement data transformations easily and consistently across various data sources. This includes filtering, reformatting, sorting, joining, merging, aggregation and other operations ready to use. The tools also support transformation scheduling, version control, monitoring and unified metadata management. Some of the ETL tools are even integrated with BI tools.

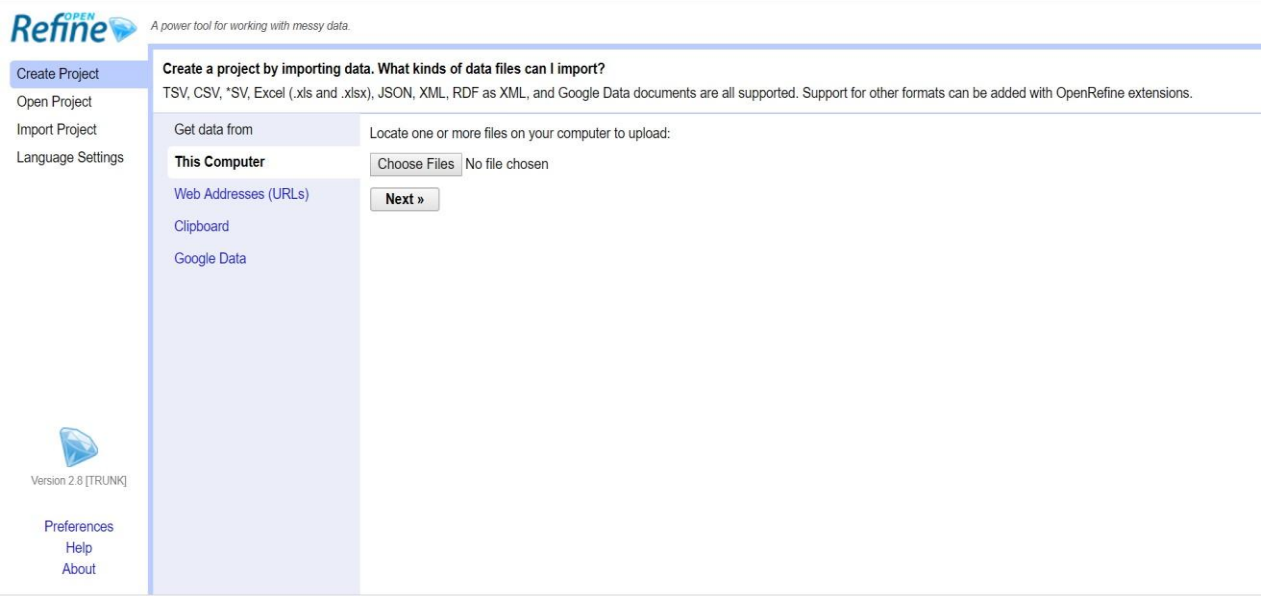
Some of the Well Known ETL Tools:

The most well-known commercial tools are Ab Initio, IBM InfoSphere DataStage, Informatica, Oracle Data Integrator, and SAP Data Integrator.

There are several open source ETL tools are **OpenRefine**,

Apatar, CloverETL, Pentaho and Talend.

In these above tools, we are going to use **OpenRefine 2.8 ETL tool**to different sample datasets forextracting, data cleaning, transforming & loading.


OpenRefine
A power tool for working with messy data.
Create Project
Open Project
Import Project
Language Settings
Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are all supported. Support for other formats can be added with OpenRefine extensions.
Get data from
This Computer
Web Addresses (URLs)
Clipboard
Google Data
Locate one or more files on your computer to upload:
Choose Files
No file chosen
Next »
Version 2.8 [TRUNK]
Preferences
Help
About

(iv). Perform various OLAP operations such slice, dice, roll up, drill down and pivot.

OLAP Operations:

Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data.

Here is the list of OLAP operations

- Roll-up (Drill-up)
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up (Drill-up):

Roll-up performs aggregation on a data cube in any of the following ways

- By climbing up a concept hierarchy for a dimension
- By dimension reduction
- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.

- When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down:

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways

- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.
- Drill-down is performed by stepping down a concept hierarchy for the dimension time.
- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.
- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

Slice:

The slice operation selects one particular dimension from a given cube and provides a new sub-cube.

Dice:

Dice selects two or more dimensions from a given cube and provides a new sub-cube.

Pivot (rotate):

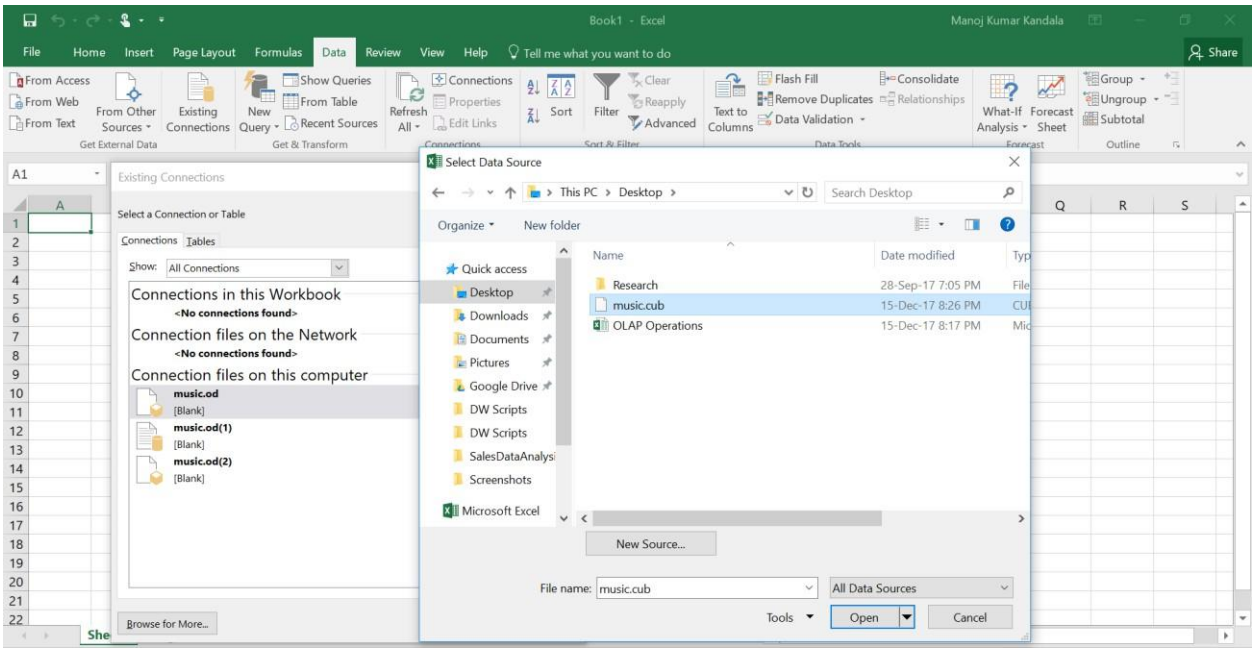
The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data.

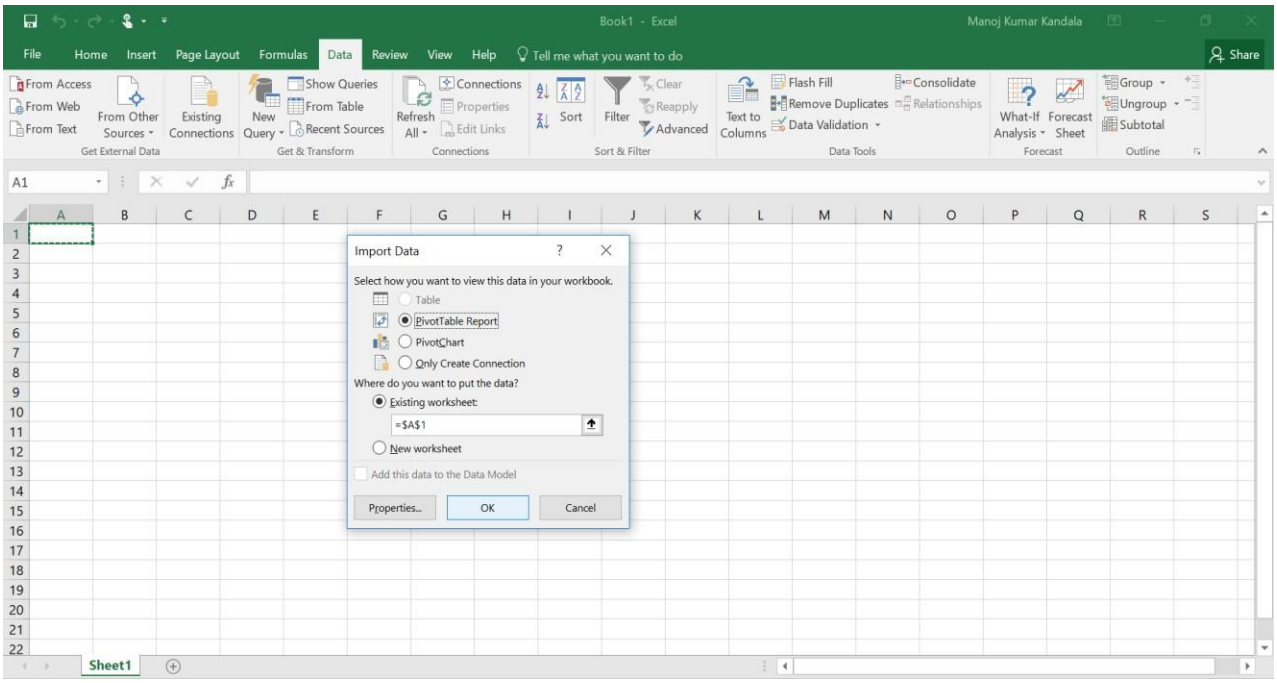
Now, we are practically implementing all these OLAP Operations using **Microsoft Excel**.

Procedure for OLAP Operations:

1. Open Microsoft Excel, go to **Data** tab in top & click on **—Existing Connections"**.

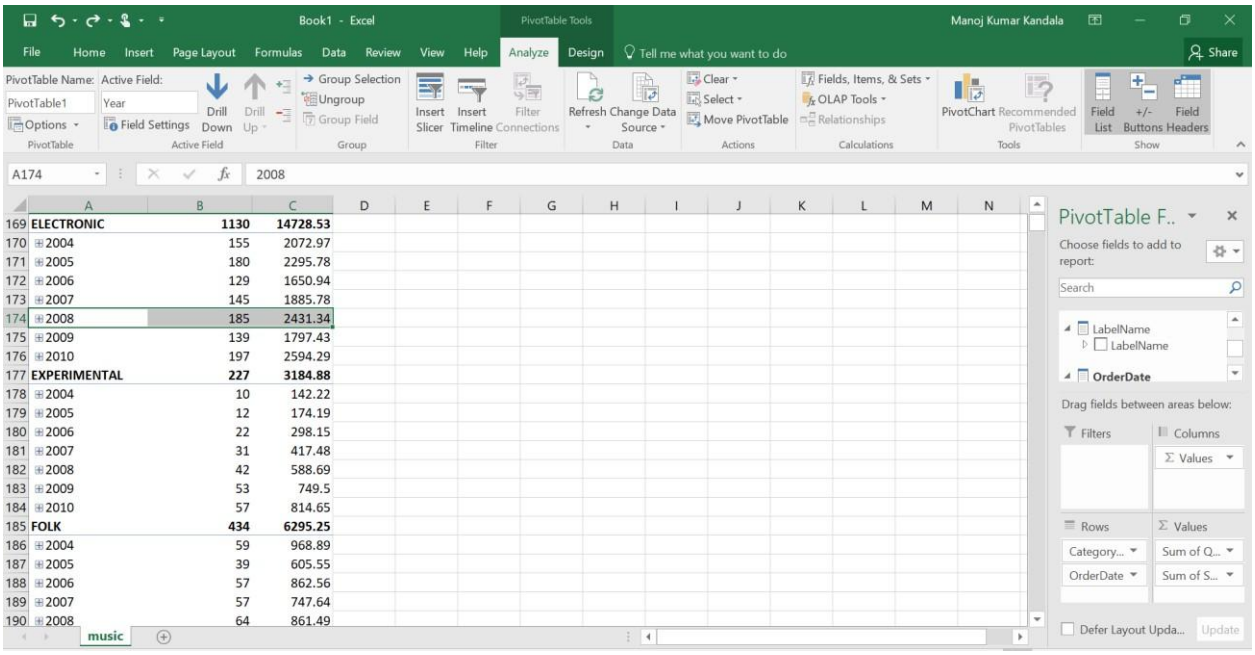
2. Existing Connections window will be opened, there **"Browse for more"** option should be clicked for importing **.cub extension file** for performing OLAP Operations. For sample, I took **music.cub** file.





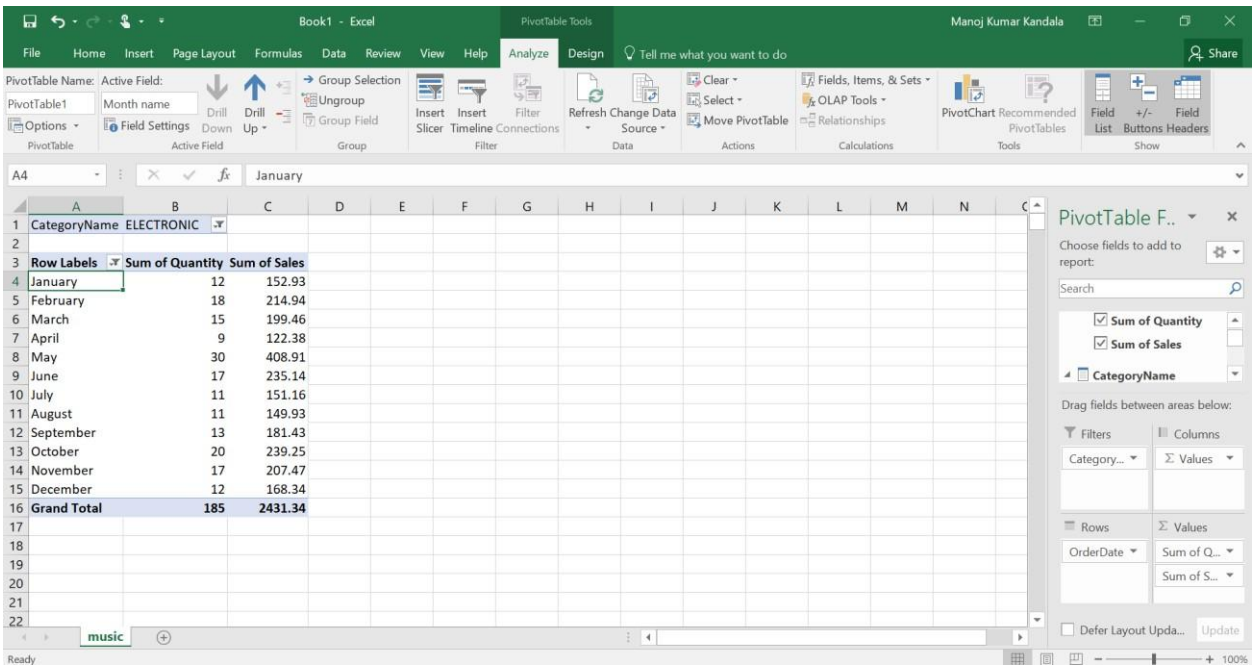
3. As shown in above window, select **—PivotTable Report"** and click **"OK".**

4. We got all the **music.cub** data for analyzing different OLAP Operations.Firstly, we performed **drill-down operation** as shown below.



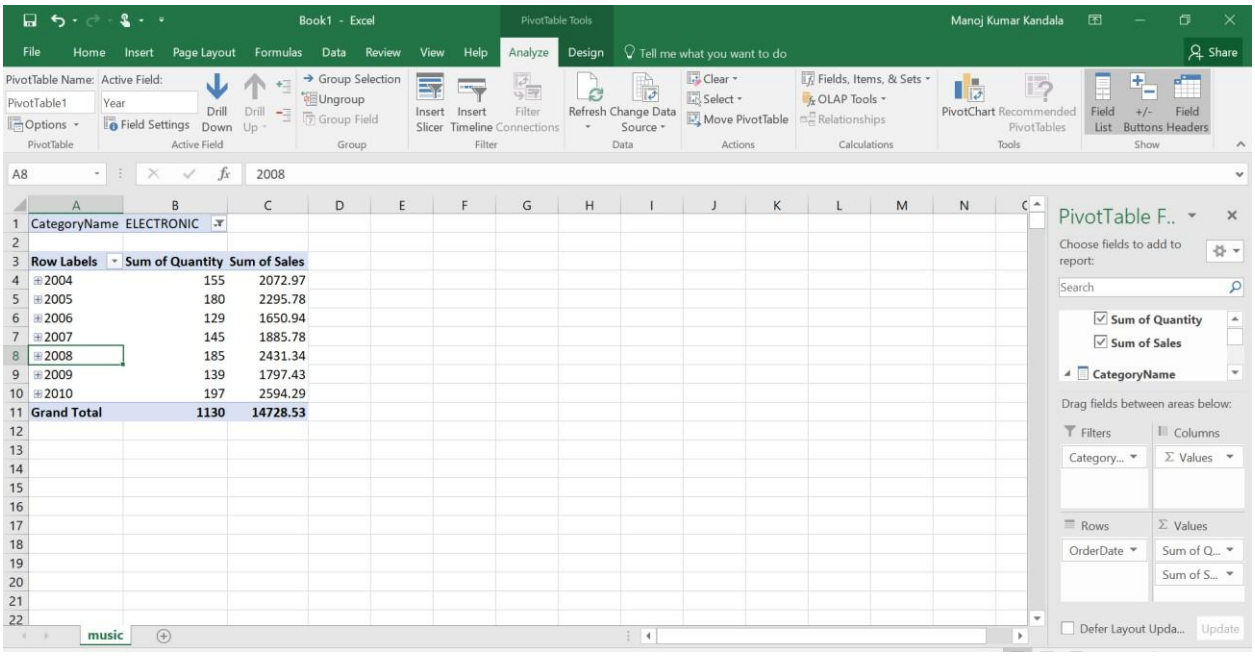
	A	B	C
169	**ELECTRONIC**	**1130**	**14728.53**
170	2004	155	2072.97
171	2005	180	2295.78
172	2006	129	1650.94
173	2007	145	1885.78
174	2008	185	2431.34
175	2009	139	1797.43
176	2010	197	2594.29
177	**EXPERIMENTAL**	**227**	**3184.88**
178	2004	10	142.22
179	2005	12	174.19
180	2006	22	298.15
181	2007	31	417.48
182	2008	42	588.69
183	2009	53	749.5
184	2010	57	814.65
185	**FOLK**	**434**	**6295.25**
186	2004	59	968.89
187	2005	39	605.55
188	2006	57	862.56
189	2007	57	747.64
190	2008	64	861.49

In the above window, we selected year **„2008"** in **„Electronic"** Category, then automatically the Drill-Down option is enabled on top navigation options. We will click on **„Drill-Down"** option, then the below window will be displayed.

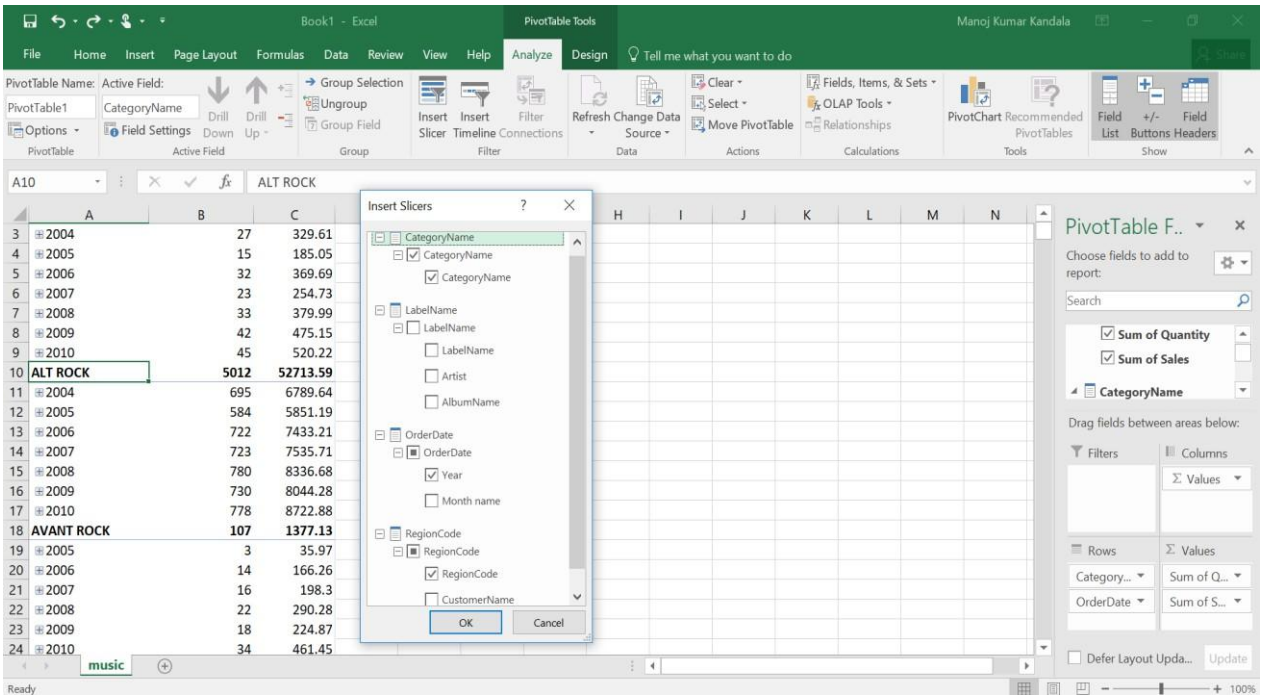


	A	B	C
1	CategoryName	ELECTRONIC	
2			
3	**Row Labels**	**Sum of Quantity**	**Sum of Sales**
4	January	12	152.93
5	February	18	214.94
6	March	15	199.46
7	April	9	122.38
8	May	30	408.91
9	June	17	235.14
10	July	11	151.16
11	August	11	149.93
12	September	13	181.43
13	October	20	239.25
14	November	17	207.47
15	December	12	168.34
16	**Grand Total**	**185**	**2431.34**

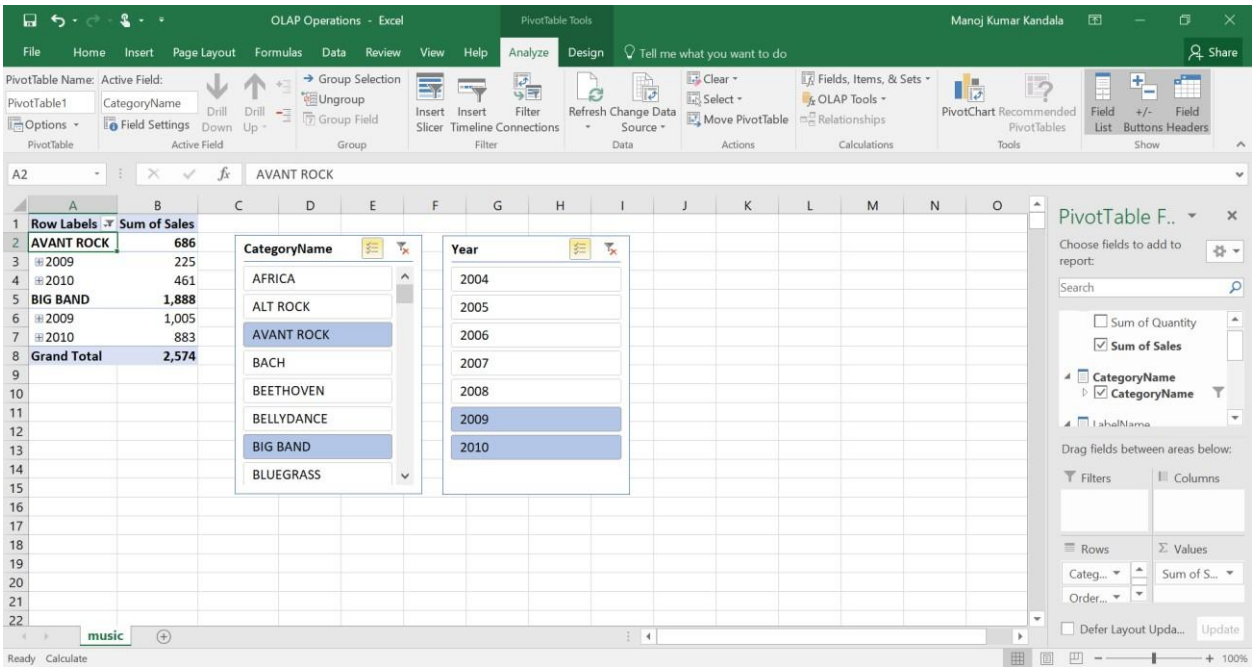
5. Now we are going to perform **roll-up (drill-up) operation**, in the above window I selected January month then automatically **Drill-up option** is enabled on top. We will click on **Drill-up** option, then the below window will be displayed.



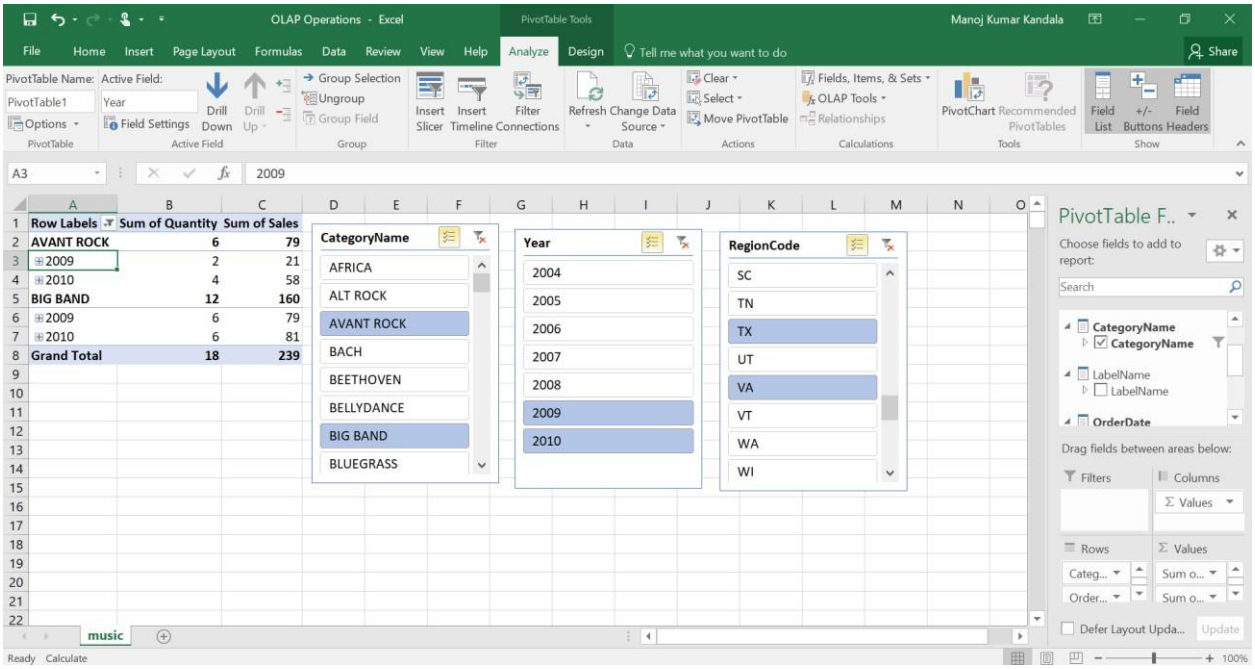
6. Next OLAP operation **Slicing** is performed by inserting slicer as shown in top navigation options.



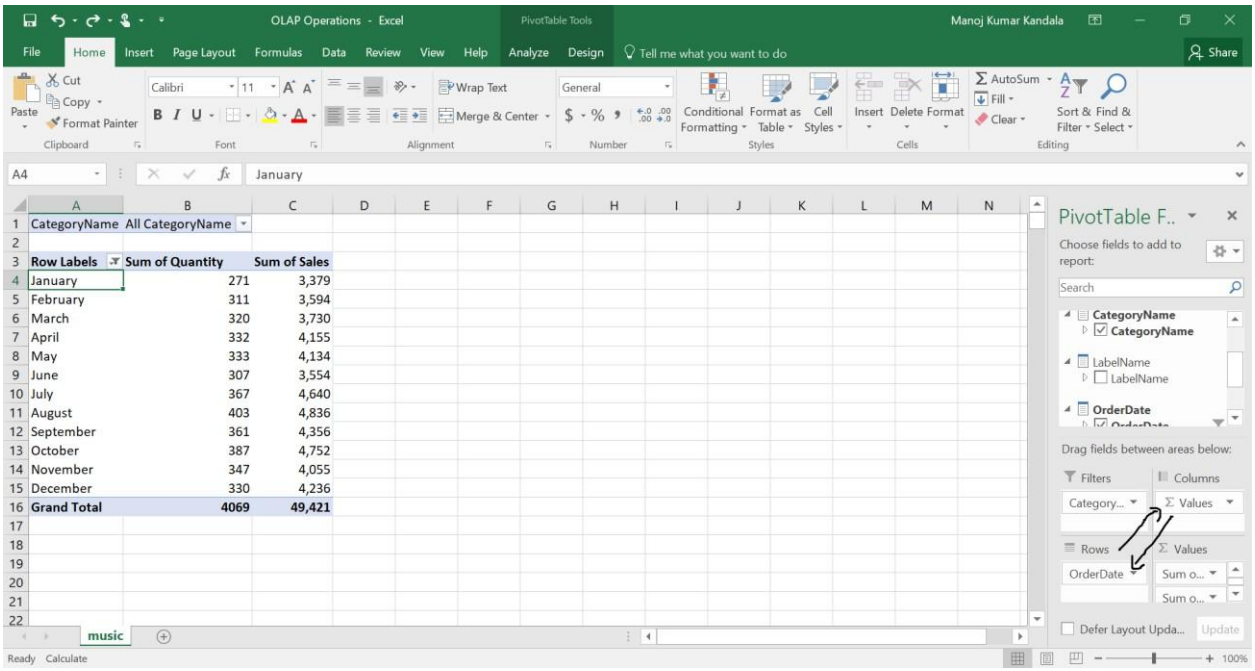
While inserting slicers for slicing operation, we select 2 Dimensions (for e.g. CategoryName & Year) only with one Measure (for e.g. Sum of sales).After inserting a slice& adding a filter (CategoryName: AVANT ROCK & BIG BAND; Year: 2009 & 2010), we will get table as shown below.



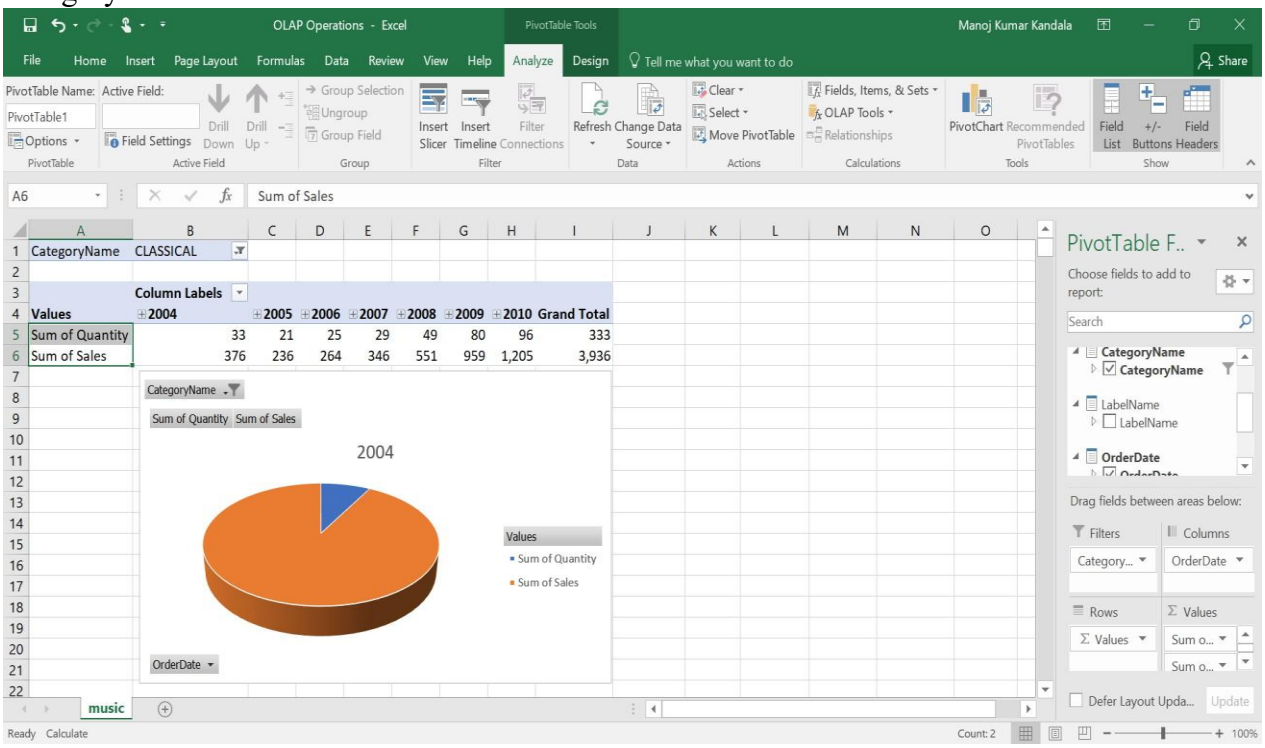
7. Dicing operation is similar to Slicing operation. Here we are selecting 3 dimensions (CategoryName, Year, RegionCode)& 2 Measures (Sum of Quantity, Sum of Sales) through **„insert slicer"** option. After that adding a filter for CategoryName, Year & RegionCode as shown below.



8. Finally, the **Pivot (rotate)** OLAP operation is performed by swapping rows (Order Date-Year) & columns (Values-Sum of Quantity & Sum of Sales) through right side bottom navigation bar as shown below.



After Swapping (rotating), we will get resultant as represented below with a pie-chart for Category-Classical& Year Wise data.



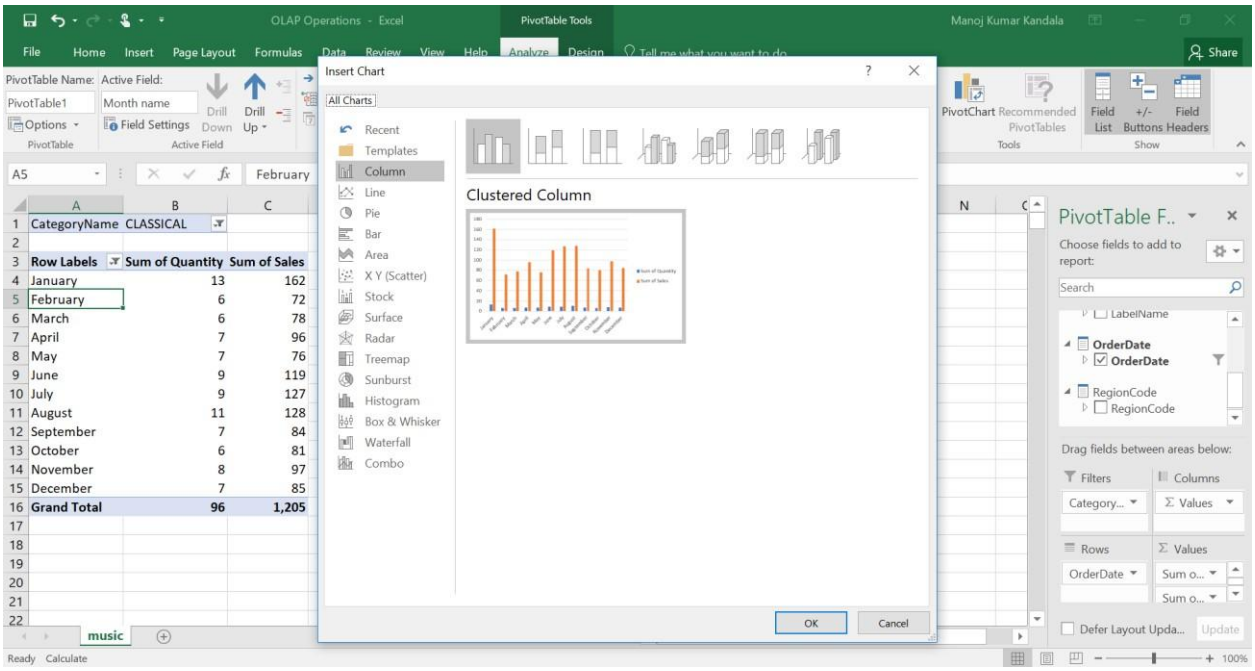
(v). Explore visualization features of the tool for analysis like identifying trends etc.

There are different visualization features for analyzing the data for trend analysisin data warehouses. Some of the popular visualizations are:

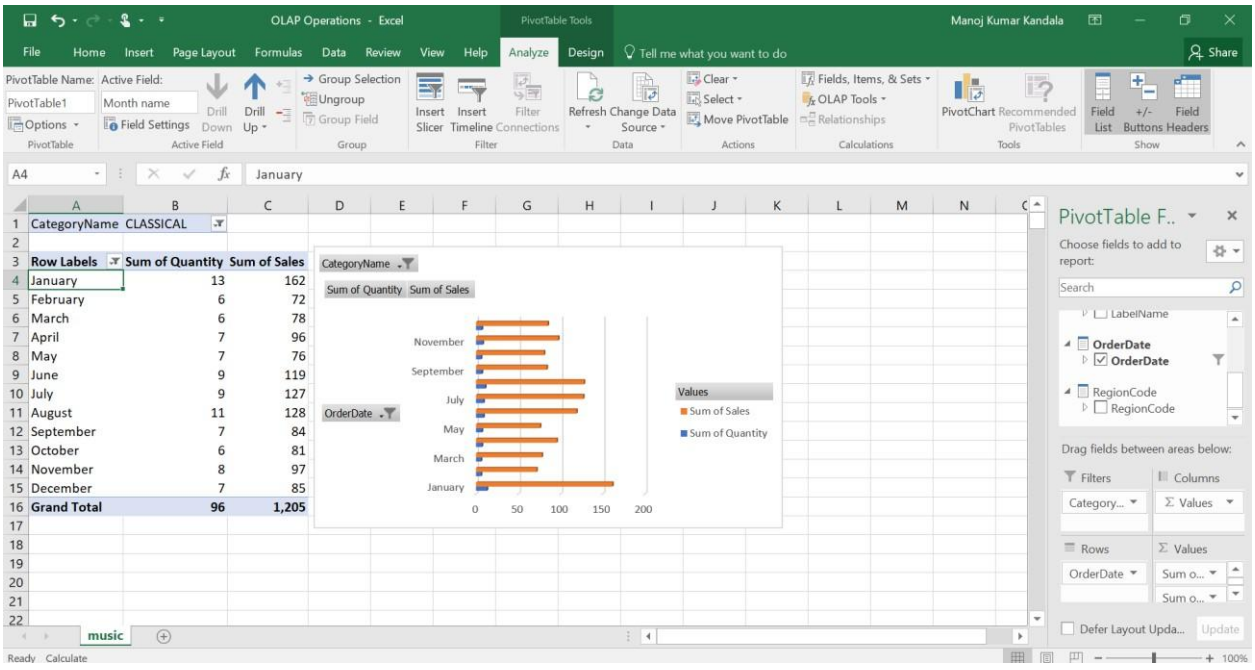
1. Column Charts

2. Line Charts
3. Pie Chart
4. Bar Graphs
4. Area Graphs
5. X & Y Scatter Graphs
6. Stock Graphs
7. Surface Charts
8. Radar Graphs
9. Treemap
10. Sunburst
11. Histogram
12. Box & Whisker
13. Waterfall
14. Combo Graphs
15. Geo Map
16. Heat Grid
17. Interactive Report
18. Stacked Column
19. Stacked Bar
20. Scatter Area

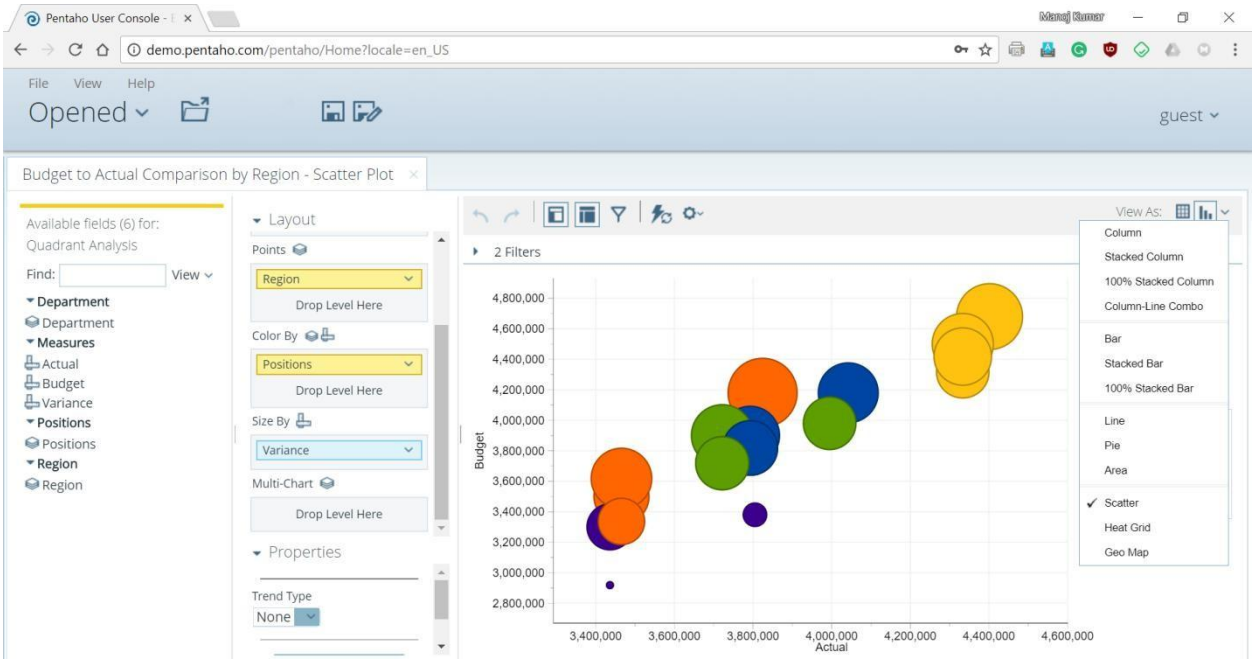
These type of visualizations can be used for analyzing data for trend analysis. Some of the tools for data visualization are Microsoft Excel, Tableau, Pentaho Business Analytics Online etc. Practically different visualization features are tested with different sample datasets.



In the below window, we used 3D-Column Charts of **Microsoft Excel** for analyzing data in data warehouse.



Below window, represents the data visualization through **Pentaho Business Analytics tool online** (http://www.pentaho.com/hosted-demo) for some sample dataset.



B. Explore WEKA Data Mining/Machine Learning Toolkit

(i). Downloading and/or installation of WEKA data mining toolkit

Procedure:

1. Go to the Weka website, http://www.cs.waikato.ac.nz/ml/weka/, and download the software. On the left-hand side, click on the link that says download.

2. Select the appropriate link corresponding to the version of the software based on your operating system and whether or not you already have Java VM running on your machine (if you don‘t know what Java VM is, then you probably don‘t).

3. The link will forward you to a site where you can download the software from a mirror site. Save the self-extracting executable to disk and then double click on it to install Weka. Answer yes or next to the questions during the installation.

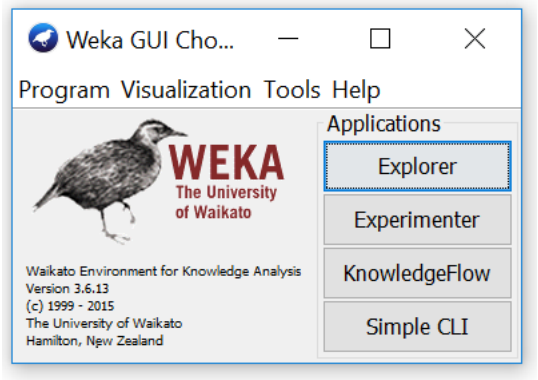
4. Click yes to accept the Java agreement if necessary. After you install the program Weka should appear on your start menu under Programs (if you are using Windows).

5. Running Weka from the start menu select Programs, then Weka.You will see the Weka GUI Chooser. Select Explorer. The Weka Explorer will then launch.

(ii). Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface, Experimenter, command-line interface.

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting pointfor launching Weka‘s main GUI applications and supporting tools. If one prefersa MDI (—multiple document interface‖) appearance, then this is provided by analternative launcher called —Main‖ (class weka.gui.Main).

The GUI Chooser consists of four buttons—one for each of the four majorWeka applications—and four menus.



The buttons can be used to start the following applications:

Explorer- An environment for exploring data with WEKA

a) Click on —explorer‖ button to bring up the explorer window.
b) Make sure the —preprocess‖ tab is highlighted.
c) Open a new file by clicking on —Open New file‖ and choosing a file with —.arff‖ extension from the —Data‖ directory.
d) Attributes appear in the window below.
e) Click on the attributes to see the visualization on the right.
f) Click —visualize all‖ to see them all

Experimenter- An environment for performing experiments and conducting statistical tests between learning schemes.

a) Experimenter is for comparing results.
b) Under the —set up‖ tab click —New‖.
c) Click on —Add New‖ under —Data‖ frame. Choose a couple of arff format files from —Data‖ directory one at a time.
d) Click on —Add New‖ under —Algorithm‖ frame. Choose several algorithms, one at a time by clicking —OK‖ in the window and —Add New‖.
e) Under the —Run‖ tab click —Start‖.

f) Wait for WEKA to finish.
g) Under ―Analyses‖ tab click on ―Experiment‖ to see results.

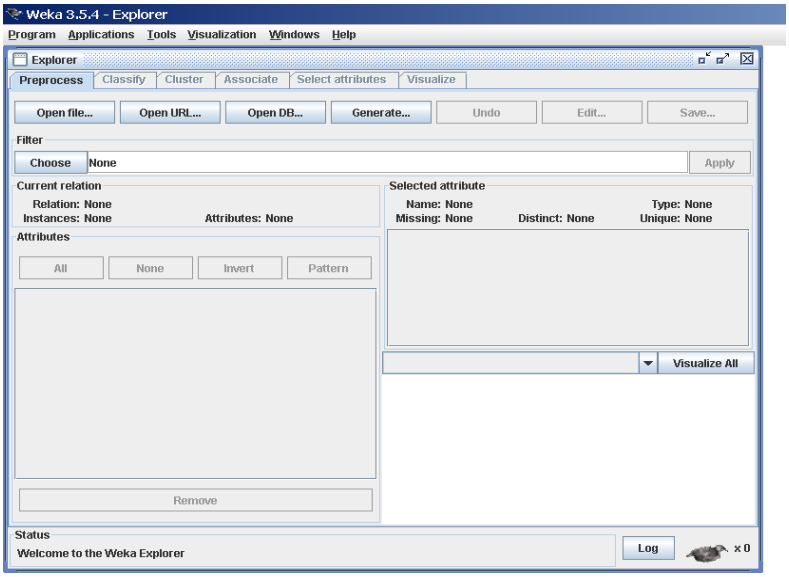
Knowledge Flow- This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantageis that it supports incremental learning.

SimpleCLI - Provides a simple command-line interface that allows directexecution of WEKA commands for operating systems that do not provide their own command line interface.

(iii). Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, classify panel, Cluster panel, Associate panel and Visualize panel)

When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.
The tabs are as follows:
1. Preprocess. Choose and modify the data being acted on.
2. Classify. Train and test learning schemes that classify or perform regression.
3. Cluster. Learn clusters for the data.
4. Associate. Learn association rules for the data.
5. Select attributes. Select the most relevant attributes in the data.
6. Visualize. View an interactive 2D plot of the data.
Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in.

1. Preprocessing



Loading Data:

The first four buttons at the top of the preprocess section enable you to loaddata into WEKA:

1. Open file.... Brings up a dialog box allowing you to browse for the datafile on the local file system.

2. Open URL .... Asks for a Uniform Resource Locator address for wherethe data is stored.

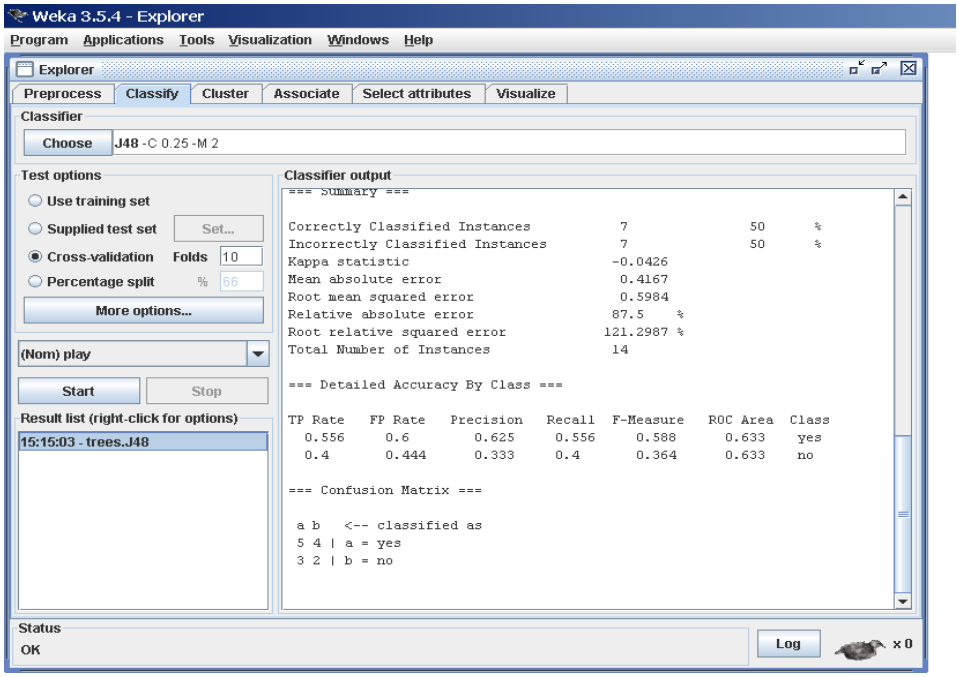
3. Open DB .... Reads data from a database. (Note that to make this workyou might have to edit the file in weka/experiment/DatabaseUtils.props.)

4. Generate. ... Enables you to generate artificial data from a variety ofDataGenerators.

Using the Open file ... button you can read files in a variety of formats:

WEKA's ARFF format, CSV format, C4.5 format, or serialized Instances format. ARFF files typically have a .arff extension, CSV files a .csv extension,C4.5 files a .data and .names extension, and serialized Instances objects a .bsiextension.

2. Classification:


Weka 3.5.4 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) play
Start Stop
Result list (right-click for options)
15:15:03 - trees.J48
Classifier output
=== Summary ===
Correctly Classified Instances 7 50 %
Incorrectly Classified Instances 7 50 %
Kappa statistic -0.0426
Mean absolute error 0.4167
Root mean squared error 0.5984
Relative absolute error 87.5 %
Root relative squared error 121.2987 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.556 0.6 0.625 0.556 0.588 0.633 yes
0.4 0.444 0.333 0.4 0.364 0.633 no
=== Confusion Matrix ===
a b <-- classified as
5 4 | a = yes
3 2 | b = no
Status
OK
Log
x 0

Selecting a Classifier

At the top of the classify section is the Classifier box. This box has a text fieldthat gives the name of the currently selected classifier, and its options. Clickingon the text box with the left mouse button brings up a GenericObjectEditordialog box, just the same as for filters, that you can use to configure the optionsof the current classifier. With a right click (or Alt+Shift+left click) you canonce again copy the setup string to the clipboard or display the properties in aGenericObjectEditor dialog box. The Choose button allows you to choose on4eof the classifiers that are available in WEKA.

Test Options

The result of applying the chosen classifier will be tested according to the optionsthat are set by clicking in the Test options box. There are four test modes:

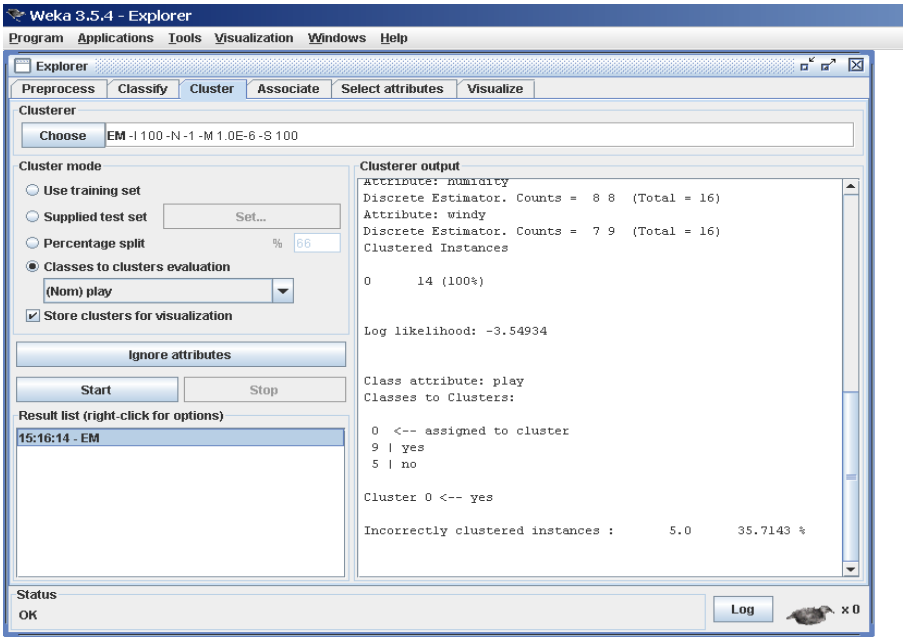
1. **Use training set:** The classifier is evaluated on how well it predicts theclass of the instances it was trained on.

2. **Supplied test set:** The classifier is evaluated on how well it predicts theclass of a set of instances loaded from a file. Clicking the Set... buttonbrings up a dialog allowing you to choose the file to test on.

3. **Cross-validation:** The classifier is evaluated by cross-validation, usingthe number of folds that are entered in the Folds text field.

4. **Percentage split:** The classifier is evaluated on how well it predicts acertain percentage of the data which is held out for testing. The amountof data held out depends on the value entered in the % field.

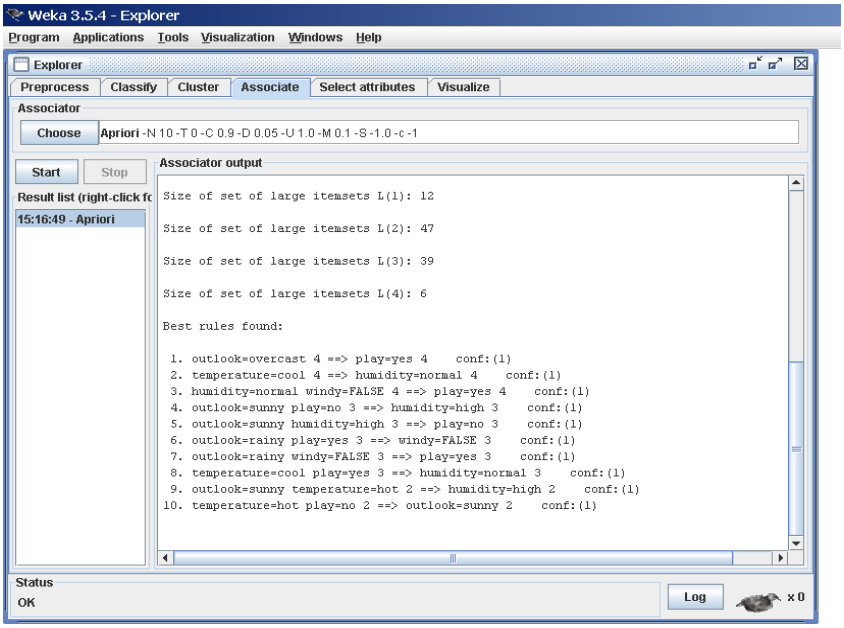
3. Clustering:



Cluster Modes:

The Cluster mode box is used to choose what to cluster and how to evaluatethe results. The first three options are the same as for classification: Use training set, Supplied test set and Percentage split.

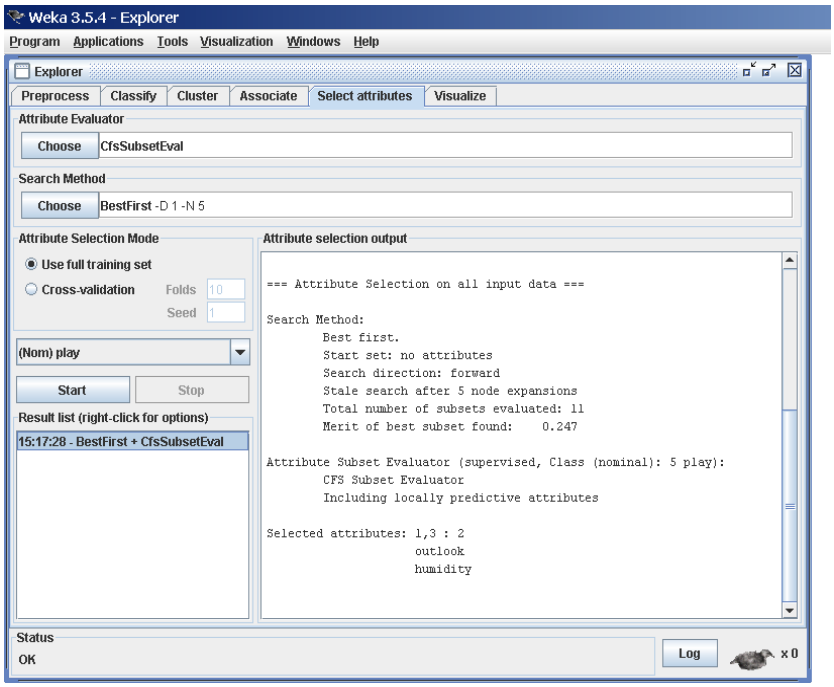
4. Associating:


Weka 3.5.4 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Associator
Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start Stop
Associator output
Result list (right-click fc
15:16:49 - Apriori
Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6
Best rules found:
1. outlook=overcast 4 ==> play=yes 4 conf:(1)
2. temperature=cool 4 ==> humidity=normal 4 conf:(1)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 conf:(1)
4. outlook=sunny play=no 3 ==> humidity=high 3 conf:(1)
5. outlook=sunny humidity=high 3 ==> play=no 3 conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 conf:(1)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 conf:(1)
8. temperature=cool play=yes 3 ==> humidity=normal 3 conf:(1)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 conf:(1)
Status
OK
Log
x 0

Setting Up

This panel contains schemes for learning association rules, and the learners are chosen and configured in the same way as the clusterers, filters, and classifiers in the other panels.

5. Selecting Attributes:

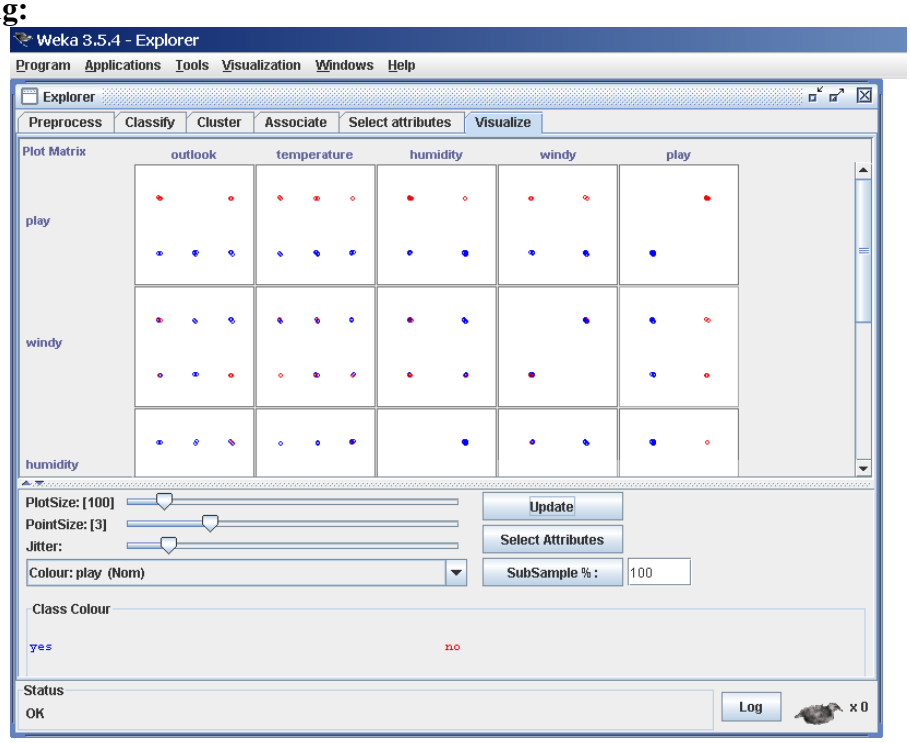


Searching and Evaluating

Attribute selection involves searching through all possible combinations of attributes in the data to find which subset of attributes works best for prediction. To do this, two objects must be set up: an attribute evaluator and a searchmethod. The evaluator determines what method is used to

assign a worth toeach subset of attributes. The search method determines what style of searchis performed.

6. Visualizing:



WEKA's visualization section allows you to visualize 2D plots of the current relation.

(iv). Study the arff file format

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software.

Overview

ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information.

The **Header** of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
```

@RELATION iris

@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

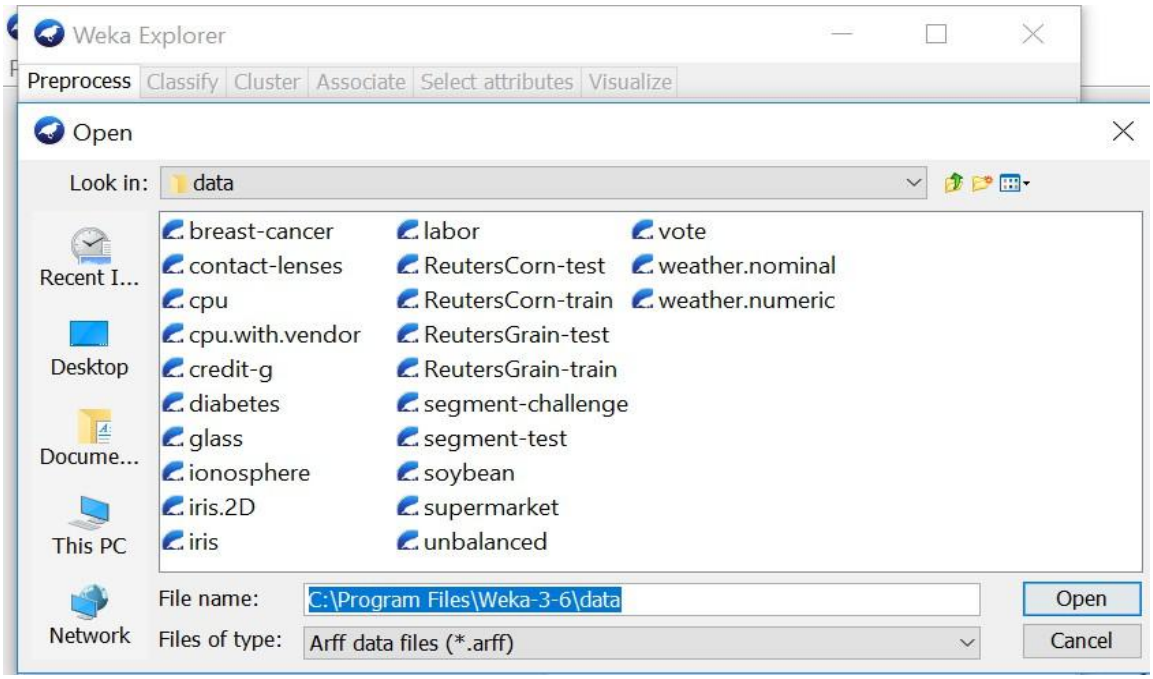
The **Data** of the ARFF file looks like the following:

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa

Lines that begin with a % are comments.
The **@RELATION**, **@ATTRIBUTE** and **@DATA** declarations are case insensitive.

(v). Explore the available data sets in WEKA

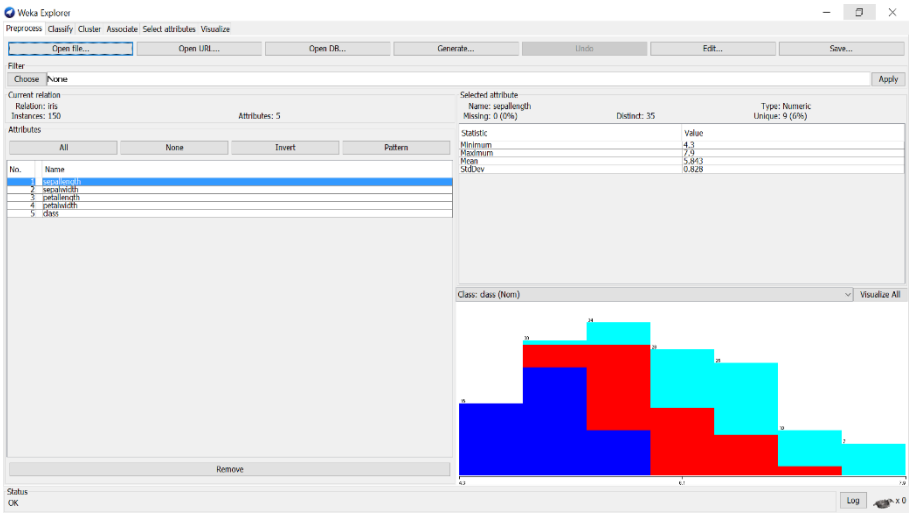
There are 23 different datasets are available in weka (C:\Program Files\Weka-3-6\) by default for testing purpose. All the datasets are available in. arff format. Those datasets are listed below.



(vi). Load a data set (ex. Weather dataset, Iris dataset, etc.)

Procedure:

1. Open the weka tool and select the explorer option.
2. New window will be opened which consists of different options (Preprocess, Association etc.)
3. In the preprocess, click the ―open file‖ option.
4. Go to C:\Program Files\Weka-3-6\data for finding different existing. arff datasets.
5. Click on any dataset for loading the data then the data will be displayed as shown below.



(vii). Load each dataset and observe the following:

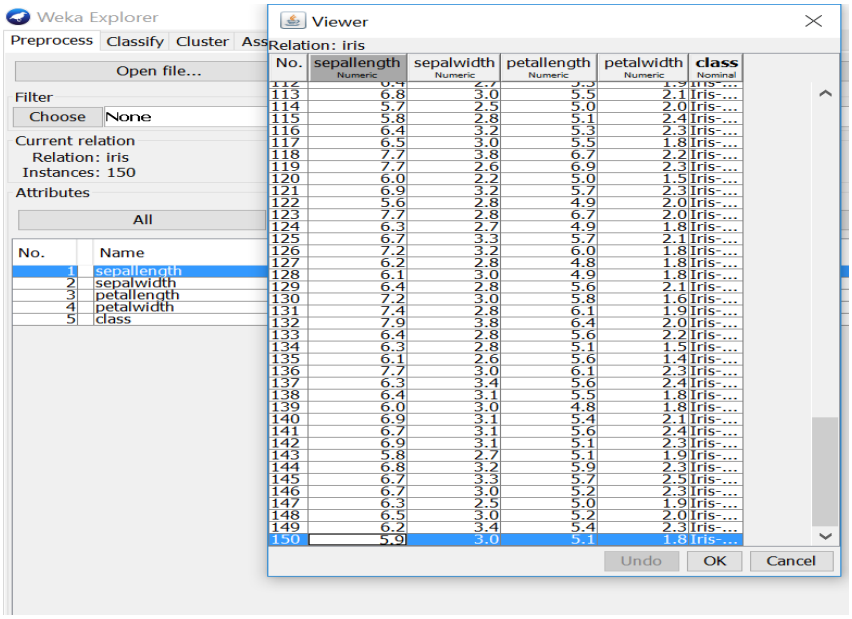
Here we have taken **IRIS.arff** dataset as sample for observing all the below things.

i. List the attribute names and they types

There are 5 attributes& its datatype present in the above loaded dataset (IRIS.arff)
sepallength – Numeric
sepalwidth – Numeric
petallength – Numeric
petallength – Numeric
Class – Nominal

ii. Number of records in each dataset

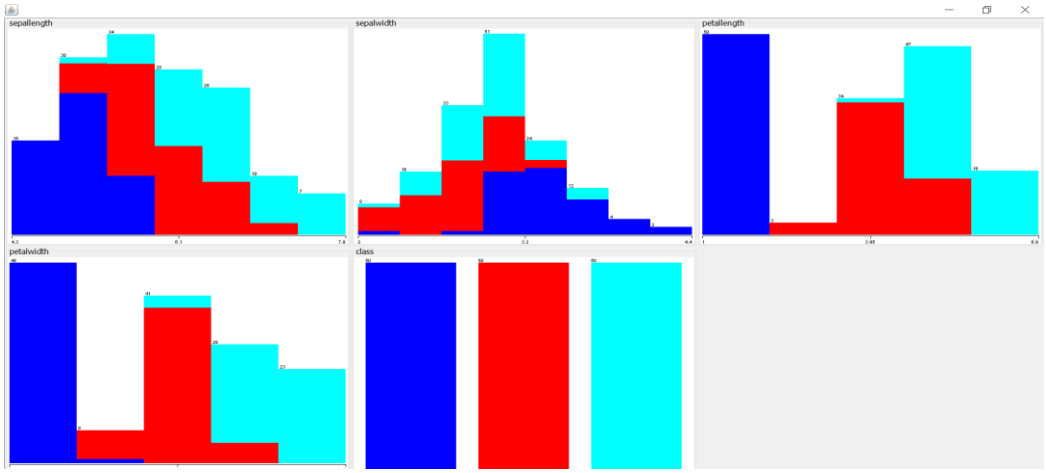
There are total <u>150</u> records (Instances) in dataset (IRIS.arff).



iii. Identify the class attribute (if any)

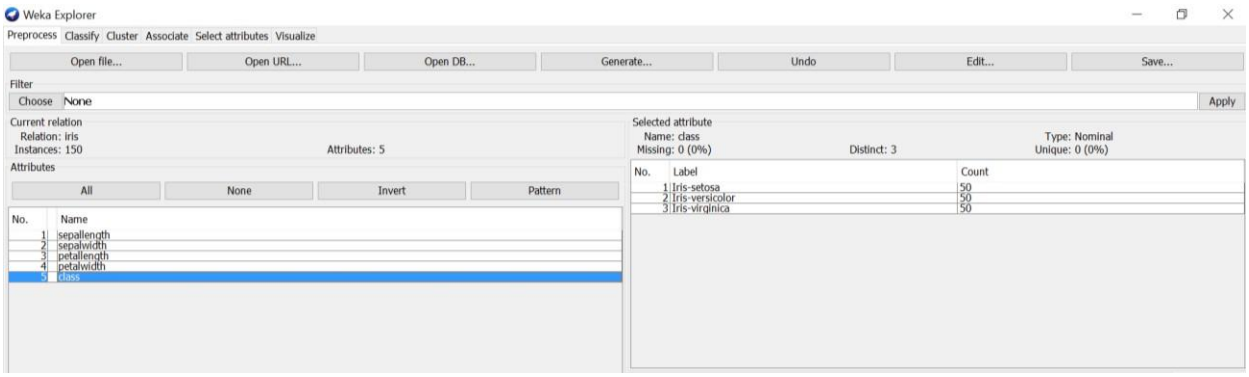
There is one class attribute which consists of 3 labels. They are:
1. Iris-setosa
2. Iris-versicolor
3. Iris-virginica

iv. Plot Histogram

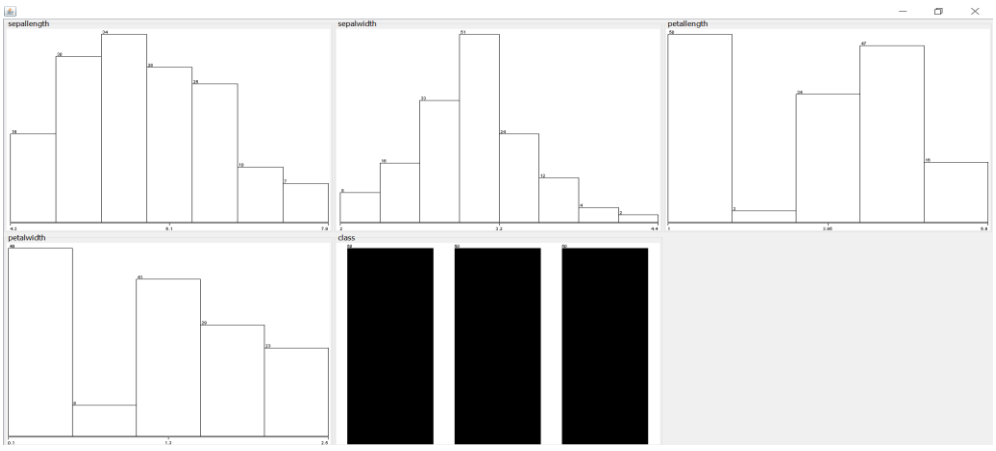


v. Determine the number of records for each class.

There is one class attribute (150 records) which consists of 3 labels. They are shown below
1. Iris-setosa - 50 records
2. Iris-versicolor – 50 records
3. Iris-virginica – 50 records



vi. Visualize the data in various dimensions

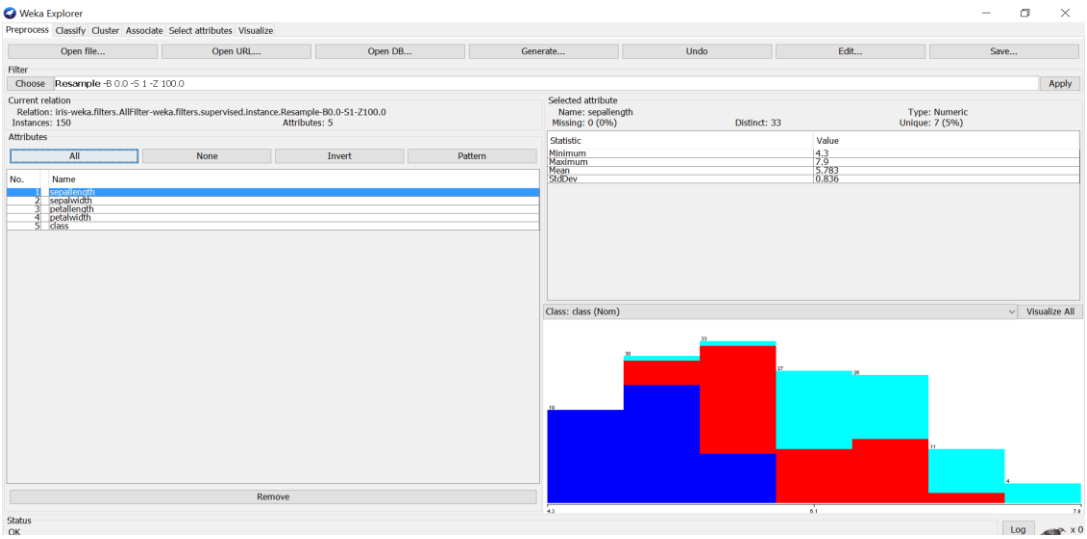


2. Perform data preprocessing tasks and Demonstrate performing association rule mining on data sets

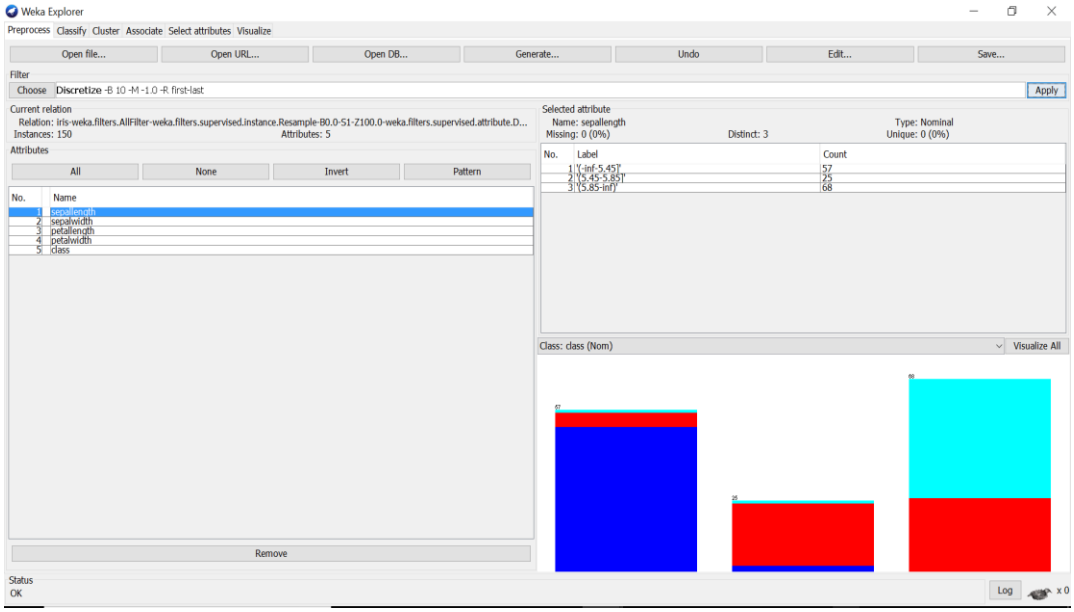
A. Explore various options available in Weka for preprocessing data and apply (like Discretization Filters, Resample filter, etc.) on each dataset

Procedure:

1. For preprocessing the data after selecting the dataset (IRIS.arff).
2. Select Filter option & apply the resample filter & see the below results.



3. Select another filter option & apply the discretization filter, see the below results



Likewise, we can apply different filters for preprocessing the data & see the results in different dimensions.

B. Load each dataset into Weka and run Aprori algorithm with different support and confidence values. Study the rules generated.

Procedure:

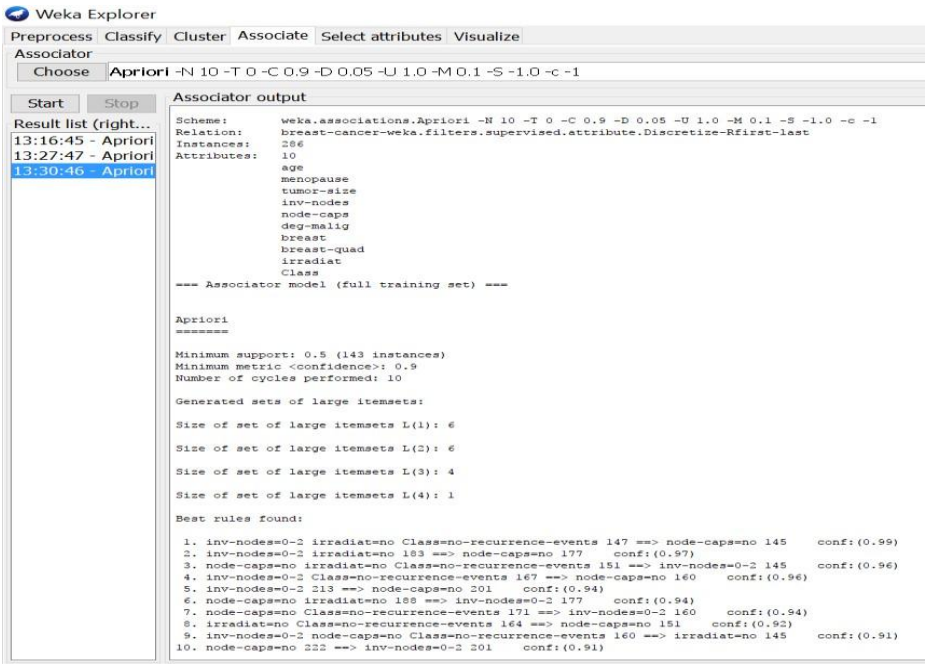
1. Load the dataset (Breast-Cancer.arff) into weka tool
2. Go to associate option & in left-hand navigation bar we can see different association algorithms.
3. In which we can select Aprori algorithm & click on select option.
4. Below we can see the rules generated with different support & confidence values for that selected dataset.

```
Relation:     breast-cancer
Instances:    286
Attributes:   10
              age
              menopause
              tumor-size
              inv-nodes
              node-caps
              deg-malig
              breast
              breast-quad
              irradiat
              Class
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.5 (143 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 6

Size of set of large itemsets L(3): 4

Size of set of large itemsets L(4): 1

Best rules found:

 1. inv-nodes=0-2 irradiat=no Class=no-recurrence-events 147 ==> node-caps=no 145    conf:(0.99)
 2. inv-nodes=0-2 irradiat=no 183 ==> node-caps=no 177    conf:(0.97)
 3. node-caps=no irradiat=no Class=no-recurrence-events 151 ==> inv-nodes=0-2 145    conf:(0.96)
 4. inv-nodes=0-2 Class=no-recurrence-events 167 ==> node-caps=no 160    conf:(0.96)
 5. inv-nodes=0-2 213 ==> node-caps=no 201    conf:(0.94)
 6. node-caps=no irradiat=no 188 ==> inv-nodes=0-2 177    conf:(0.94)
 7. node-caps=no Class=no-recurrence-events 171 ==> inv-nodes=0-2 160    conf:(0.94)
 8. irradiat=no Class=no-recurrence-events 164 ==> node-caps=no 151    conf:(0.92)
 9. inv-nodes=0-2 node-caps=no Class=no-recurrence-events 160 ==> irradiat=no 145    conf:(0.91)
10. node-caps=no 222 ==> inv-nodes=0-2 201    conf:(0.91)
```

C. Apply different discretization filters on numerical attributes and run the Apriori association rule algorithm. Study the rules generated. Derive interesting insights and observe the effect of discretization in the rule generation process.

Procedure:

1. Load the dataset (Breast-Cancer.arff) into weka tool& select the discretize filter & apply it.
2. Go to associate option & in left-hand navigation bar we can see different association algorithms.
3. In which we can select Aprori algorithm & click on select option.
4. Below we can see the rules generated with different support & confidence values for that selected dataset.

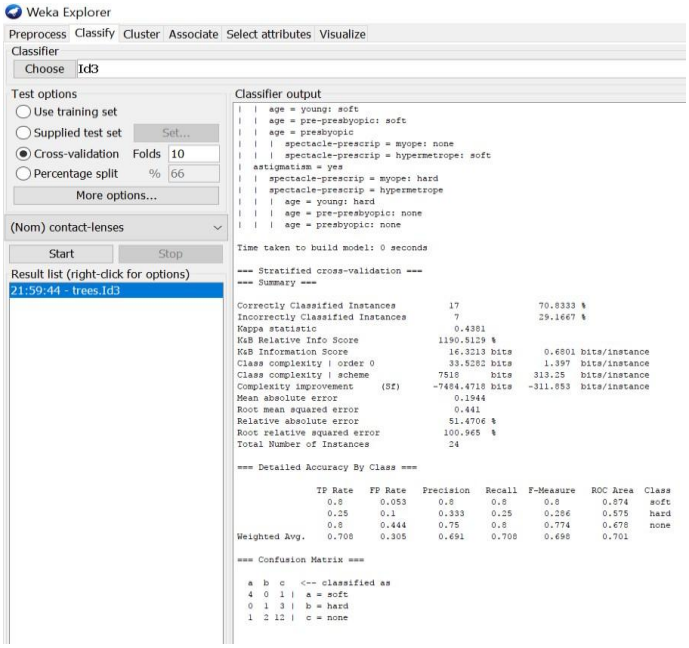

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right...
13:16:45 - Apriori
13:27:47 - Apriori
13:30:46 - Apriori
Associator output
Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: breast-cancer-weka.filters.supervised.attribute.Discretize-Rfirst-last
Instances: 286
Attributes: 10
age
menopause
tumor-size
inv-nodes
node-caps
deg-malig
breast
breast-quad
irradiat
Class
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.5 (143 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 10
Generated sets of large itemsets:
Size of set of large itemsets L(1): 6
Size of set of large itemsets L(2): 6
Size of set of large itemsets L(3): 4
Size of set of large itemsets L(4): 1
Best rules found:
1. inv-nodes=0-2 irradiat=no Class=no-recurrence-events 147 ==> node-caps=no 145 conf:(0.99)
2. inv-nodes=0-2 irradiat=no 183 ==> node-caps=no 177 conf:(0.97)
3. node-caps=no irradiat=no Class=no-recurrence-events 151 ==> inv-nodes=0-2 145 conf:(0.96)
4. inv-nodes=0-2 Class=no-recurrence-events 167 ==> node-caps=no 160 conf:(0.96)
5. inv-nodes=0-2 213 ==> node-caps=no 201 conf:(0.94)
6. node-caps=no irradiat=no 188 ==> inv-nodes=0-2 177 conf:(0.94)
7. node-caps=no Class=no-recurrence-events 171 ==> inv-nodes=0-2 160 conf:(0.94)
8. irradiat=no Class=no-recurrence-events 164 ==> node-caps=no 151 conf:(0.92)
9. inv-nodes=0-2 node-caps=no Class=no-recurrence-events 160 ==> irradiat=no 145 conf:(0.91)
10. node-caps=no 222 ==> inv-nodes=0-2 201 conf:(0.91)

3. Demonstrate performing classification on data sets

A. Load each dataset into Weka and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic.

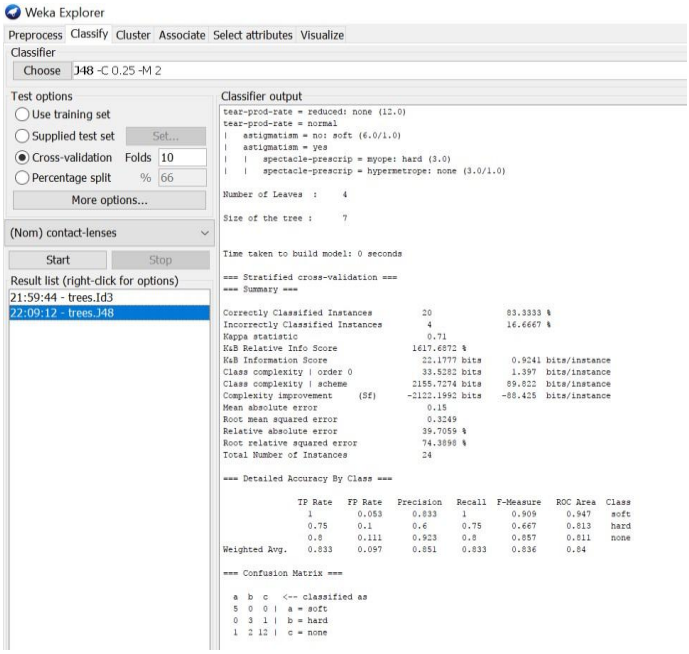
Procedure for Id3:

1. Load the dataset (Contact-lenses.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under tree section.
3. In which we selected Id3 algorithm, in more options select the output entropy evaluation measures& click on start option.
4. Then we will get classifier output, entropy values& Kappa Statistic as represented below.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose Id3
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) contact-lenses
Start Stop
Result list (right-click for options)
21:59:44 - trees.Id3
Classifier output
| | age = young: soft
| | age = pre-presbyopic: soft
| | age = presbyopic
| | | spectacle-prescrip = myope: none
| | | spectacle-prescrip = hypermetrope: soft
| astigmatism = yes
| | spectacle-prescrip = myope: hard
| | spectacle-prescrip = hypermetrope
| | | age = young: hard
| | | age = pre-presbyopic: none
| | | age = presbyopic: none
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 17 70.8333 %
Incorrectly Classified Instances 7 29.1667 %
Kappa statistic 0.4381
K&B Relative Info Score 1190.5129 %
K&B Information Score 16.3213 bits 0.6801 bits/instance
Class complexity | order 0 33.5282 bits 1.397 bits/instance
Class complexity | scheme 7518 bits 313.25 bits/instance
Complexity improvement (Sf) -7484.4718 bits -311.853 bits/instance
Mean absolute error 0.1944
Root mean squared error 0.441
Relative absolute error 51.4706 %
Root relative squared error 100.965 %
Total Number of Instances 24
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.8 0.053 0.8 0.8 0.8 0.874 soft
0.25 0.1 0.333 0.25 0.286 0.575 hard
0.8 0.444 0.75 0.8 0.774 0.678 none
Weighted Avg. 0.708 0.305 0.691 0.708 0.698 0.701
=== Confusion Matrix ===
a b c <-- classified as
4 0 1 | a = soft
0 1 3 | b = hard
1 2 12 | c = none

5. In the above screenshot, we can run classifiers with different test options (Cross-validation, Use Training Set, Percentage Split, Supplied Test set).
The result of applying the chosen classifier will be tested according to the optionsthat are set by clicking in the Test options box. There are four test modes:
A. Use training set: The classifier is evaluated on how well it predicts theclass of the instances it was trained on.
B. Supplied test set: The classifier is evaluated on how well it predicts theclass of a set ofinstances loaded from a file. Clicking the Set... buttonbrings up a dialog allowing you to choose the file to test on.
C. Cross-validation: The classifier is evaluated by cross-validation, usingthe number of folds that are entered in the Folds text field.
D. Percentage split: The classifier is evaluated on how well it predicts acertain percentage of the data which is held out for testing. The amountof data held out depends on the value entered in the % field.

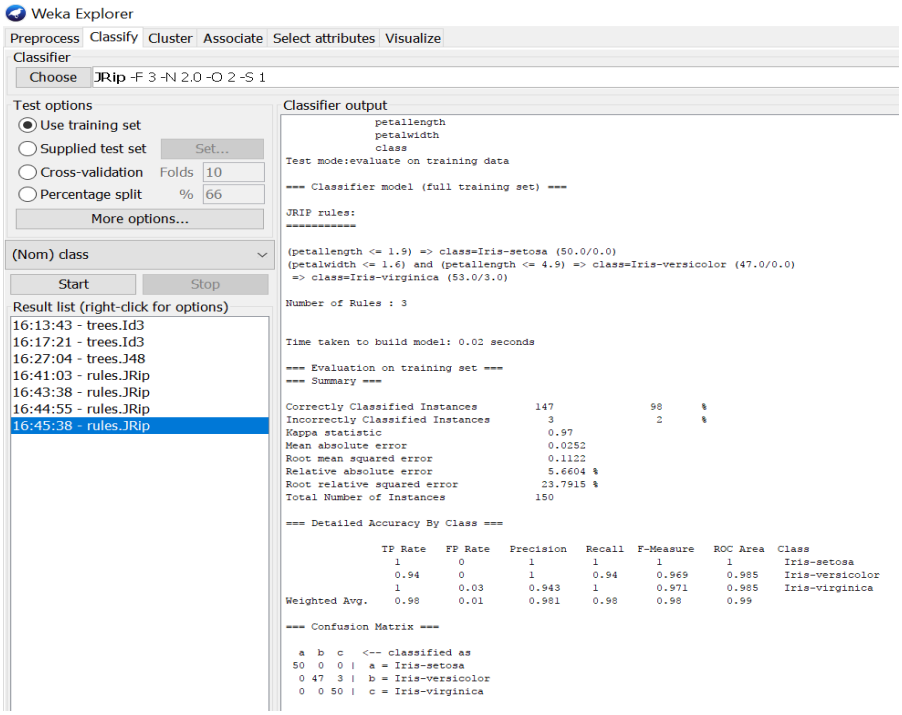
Procedure for J48:
1. Load the dataset (Contact-lenses.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under tree section.
3. In which we selected J48 algorithm,in more options select the output entropy evaluation measures& click on start option.
4. Then we will get classifier output, entropy values & Kappa Statistic as represented below.
5. In the below screenshot, we can run classifiers with different test options (Cross-validation, Use Training Set, Percentage Split, Supplied Test set).


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) contact-lenses
Start Stop
Result list (right-dick for options)
21:59:44 - trees.Id3
22:09:12 - trees.J48
Classifier output
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
| astigmatism = no: soft (6.0/1.0)
| astigmatism = yes
| | spectacle-prescrip = myope: hard (3.0)
| | spectacle-prescrip = hypermetrope: none (3.0/1.0)
Number of Leaves : 4
Size of the tree : 7
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 20 83.3333 %
Incorrectly Classified Instances 4 16.6667 %
Kappa statistic 0.71
K&B Relative Info Score 1617.6872 %
K&B Information Score 22.1777 bits 0.9241 bits/instance
Class complexity | order 0 33.5282 bits 1.397 bits/instance
Class complexity | scheme 2155.7274 bits 89.822 bits/instance
Complexity improvement (Sf) -2122.1992 bits -88.425 bits/instance
Mean absolute error 0.15
Root mean squared error 0.3249
Relative absolute error 39.7059 %
Root relative squared error 74.3898 %
Total Number of Instances 24
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0.053 0.833 1 0.909 0.947 soft
0.75 0.1 0.6 0.75 0.667 0.813 hard
0.8 0.111 0.923 0.8 0.857 0.811 none
Weighted Avg. 0.833 0.097 0.851 0.833 0.836 0.84
=== Confusion Matrix ===
a b c <-- classified as
5 0 0 | a = soft
0 3 1 | b = hard
1 2 12 | c = none

B. Extract if-then rules from the decision tree generated by the classifier, Observethe confusion matrix and derive Accuracy, F-measure, TPrate, FPrate, Precision and Recall values. Apply cross-validation strategy with various fold levels and compare the accuracy results.

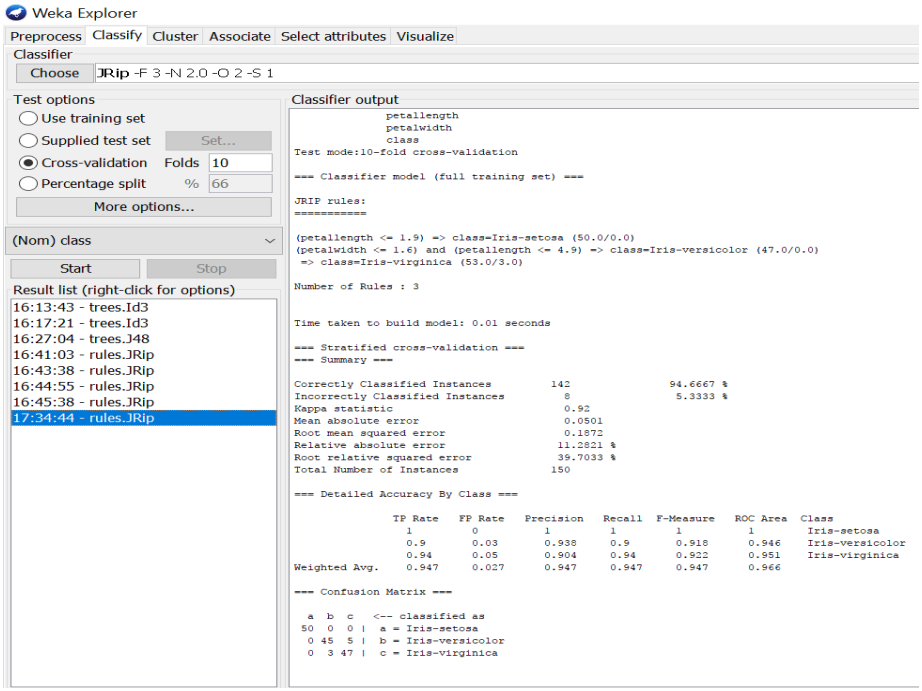
Procedure:

1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under rules section.
3. In which we selected JRip (If-then) algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists ofF-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose JRip -F 3 -N 2.0 -O 2 -S 1
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) class
Start Stop
Result list (right-click for options)
16:13:43 - trees.Id3
16:17:21 - trees.Id3
16:27:04 - trees.J48
16:41:03 - rules.JRip
16:43:38 - rules.JRip
16:44:55 - rules.JRip
16:45:38 - rules.JRip
Classifier output
petallength
petalwidth
class
Test mode:evaluate on training data
=== Classifier model (full training set) ===
JRIP rules:
(petallength <= 1.9) => class=Iris-setosa (50.0/0.0)
(petalwidth <= 1.6) and (petallength <= 4.9) => class=Iris-versicolor (47.0/0.0)
=> class=Iris-virginica (53.0/3.0)
Number of Rules : 3
Time taken to build model: 0.02 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 147 98 %
Incorrectly Classified Instances 3 2 %
Kappa statistic 0.97
Mean absolute error 0.0252
Root mean squared error 0.1122
Relative absolute error 5.6604 %
Root relative squared error 23.7915 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 Iris-setosa
0.94 0 1 0.94 0.969 0.985 Iris-versicolor
1 0.03 0.943 1 0.971 0.985 Iris-virginica
Weighted Avg. 0.98 0.01 0.981 0.98 0.98 0.99
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 0 50 | c = Iris-virginica

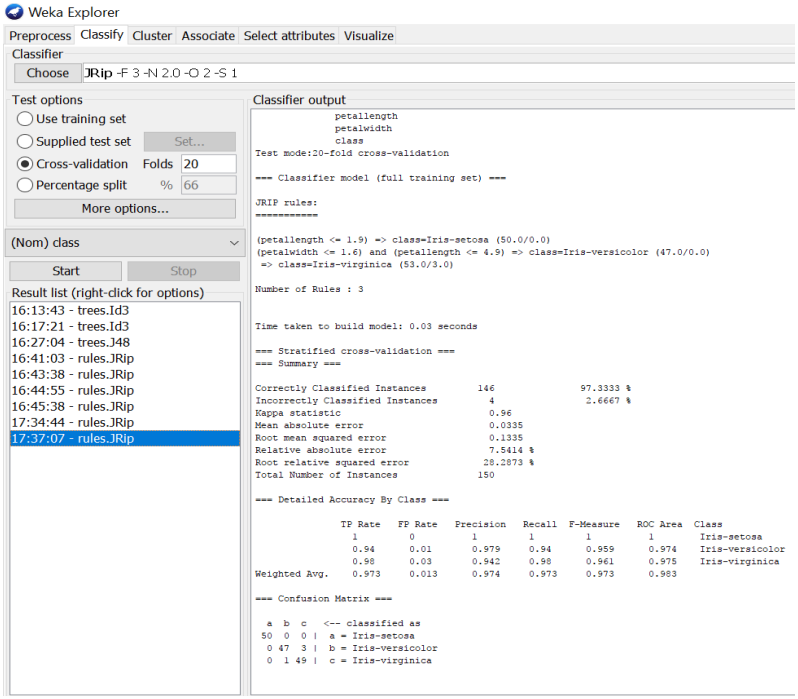
Using Cross-Validation Strategy with 10 folds:

Here, we enabled cross-validation test option with 10 folds & clicked start button as represented below.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose JRip -F 3 -N 2.0 -O 2 -S 1
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) class
Start Stop
Result list (right-click for options)
16:13:43 - trees.Id3
16:17:21 - trees.Id3
16:27:04 - trees.J48
16:41:03 - rules.JRip
16:43:38 - rules.JRip
16:44:55 - rules.JRip
16:45:38 - rules.JRip
17:34:44 - rules.JRip
Classifier output
petallength
petalwidth
class
Test mode:10-fold cross-validation
=== Classifier model (full training set) ===
JRIP rules:
(petallength <= 1.9) => class=Iris-setosa (50.0/0.0)
(petalwidth <= 1.6) and (petallength <= 4.9) => class=Iris-versicolor (47.0/0.0)
=> class=Iris-virginica (53.0/3.0)
Number of Rules : 3
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 142 94.6667 %
Incorrectly Classified Instances 8 5.3333 %
Kappa statistic 0.92
Mean absolute error 0.0501
Root mean squared error 0.1872
Relative absolute error 11.2821 %
Root relative squared error 39.7033 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 Iris-setosa
0.9 0.03 0.938 0.9 0.918 0.946 Iris-versicolor
0.94 0.05 0.904 0.94 0.922 0.951 Iris-virginica
Weighted Avg. 0.947 0.027 0.947 0.947 0.947 0.966
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 45 5 | b = Iris-versicolor
0 3 47 | c = Iris-virginica

Using Cross-Validation Strategy with 20 folds:

Here, we enabled cross-validation test option with 20 folds & clicked start button as represented below.

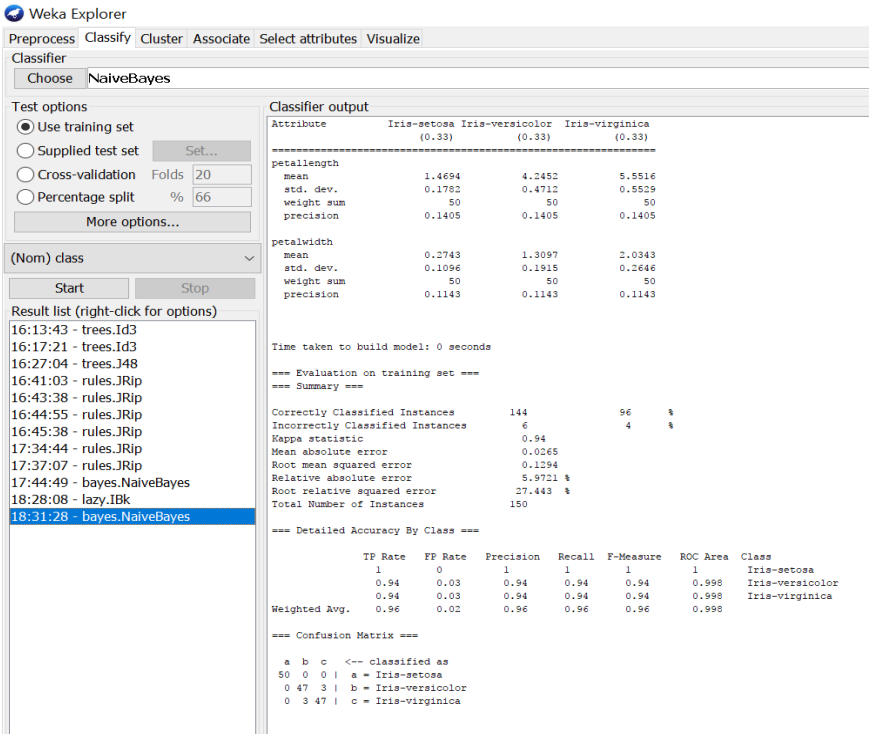

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose JRip -F 3 -N 2.0 -O 2 -S 1
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 20
Percentage split % 66
More options...
(Nom) class
Start Stop
Result list (right-click for options)
16:13:43 - trees.Id3
16:17:21 - trees.Id3
16:27:04 - trees.J48
16:41:03 - rules.JRip
16:43:38 - rules.JRip
16:44:55 - rules.JRip
16:45:38 - rules.JRip
17:34:44 - rules.JRip
17:37:07 - rules.JRip
Classifier output
petallength
petalwidth
class
Test mode:20-fold cross-validation
=== Classifier model (full training set) ===
JRIP rules:
(petallength <= 1.9) => class=Iris-setosa (50.0/0.0)
(petalwidth <= 1.6) and (petallength <= 4.9) => class=Iris-versicolor (47.0/0.0)
=> class=Iris-virginica (53.0/3.0)
Number of Rules : 3
Time taken to build model: 0.03 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 146 97.3333 %
Incorrectly Classified Instances 4 2.6667 %
Kappa statistic 0.96
Mean absolute error 0.0335
Root mean squared error 0.1335
Relative absolute error 7.5414 %
Root relative squared error 28.2873 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 Iris-setosa
0.94 0.01 0.979 0.94 0.959 0.974 Iris-versicolor
0.98 0.03 0.942 0.98 0.961 0.975 Iris-virginica
Weighted Avg. 0.973 0.013 0.974 0.973 0.973 0.983
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 1 49 | c = Iris-virginica

If we see the above results of cross validation with 10 folds & 20 folds. As per our observation the error rate is lesser with 20 folds got 97.3% correctness when compared to 10 folds got 94.6% correctness.

C. Load each dataset into Weka and perform Naive-bayes classification and k-Nearest Neighbour classification. Interpret the results obtained.

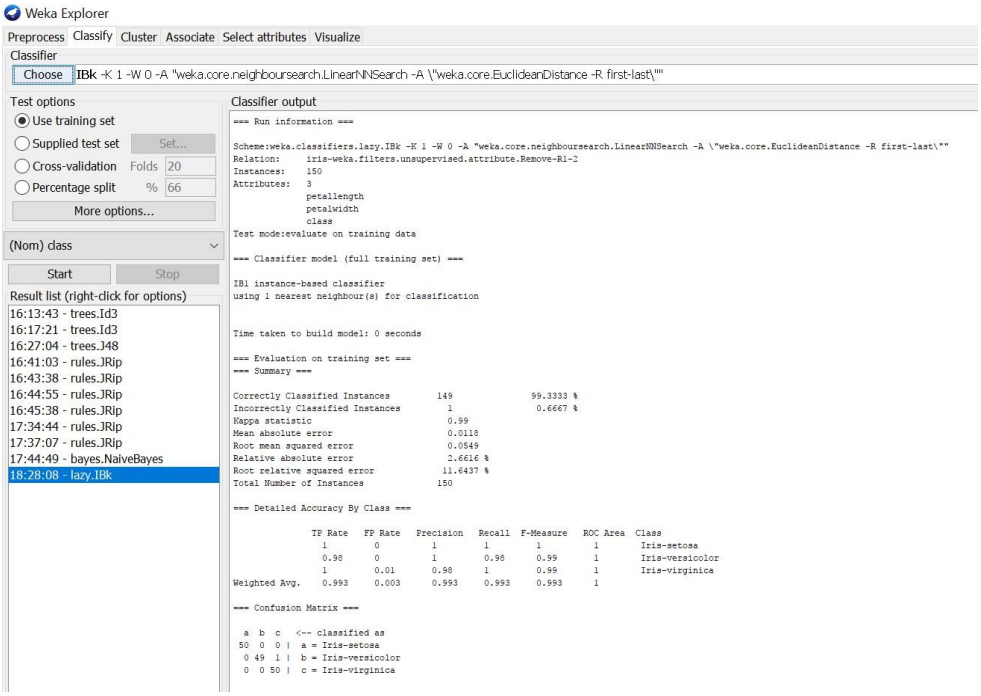
Procedure for Naïve-Bayes:

1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under bayes section.
3. In which we selected Naïve-Bayes algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.



Procedure for K-Nearest Neighbour (IBK):

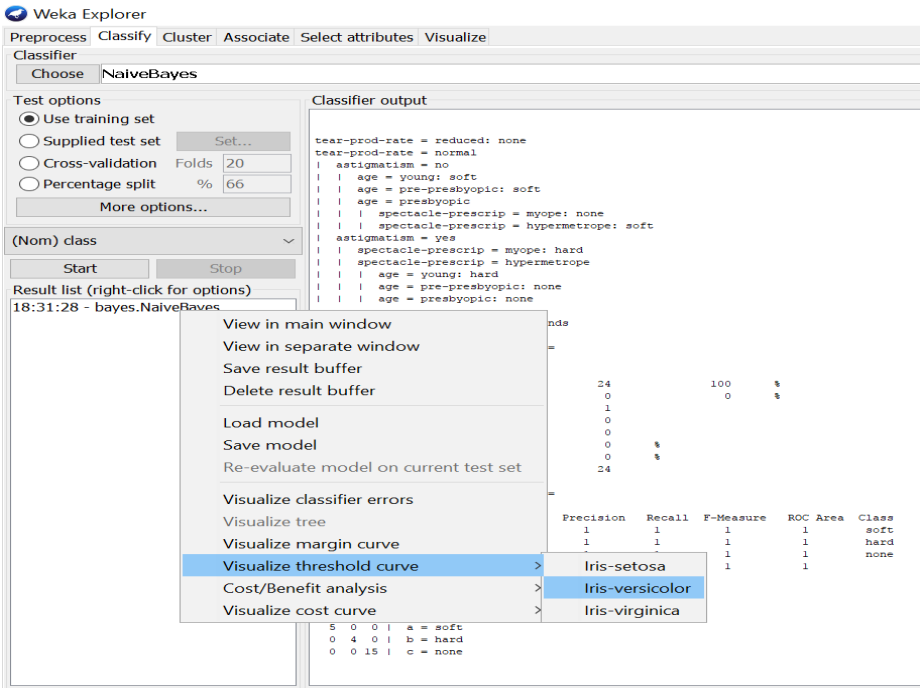
1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under lazy section.
3. In which we selected K-Nearest Neighbour (IBK) algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.

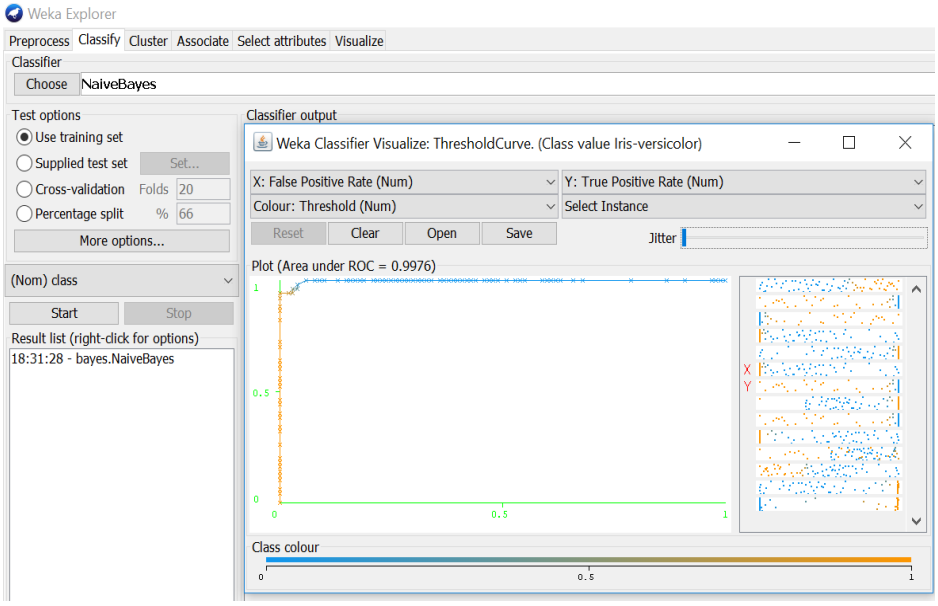

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 20
Percentage split % 66
More options...
(Nom) class
Start Stop
Result list (right-click for options)
16:13:43 - trees.Id3
16:17:21 - trees.Id3
16:27:04 - trees.J48
16:41:03 - rules.JRip
16:43:38 - rules.JRip
16:44:55 - rules.JRip
16:45:38 - rules.JRip
17:34:44 - rules.JRip
17:37:07 - rules.JRip
17:44:49 - bayes.NaiveBayes
18:28:08 - lazy.IBk
Classifier output
=== Run information ===
Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation: iris-weka.filters.unsupervised.attribute.Remove-R1-2
Instances: 150
Attributes: 3
petallength
petalwidth
class
Test mode:evaluate on training data
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 149 99.3333 %
Incorrectly Classified Instances 1 0.6667 %
Kappa statistic 0.99
Mean absolute error 0.0118
Root mean squared error 0.0549
Relative absolute error 2.6616 %
Root relative squared error 11.6437 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 Iris-setosa
0.98 0 1 0.98 0.99 1 Iris-versicolor
1 0.01 0.98 1 0.99 1 Iris-virginica
Weighted Avg. 0.993 0.003 0.993 0.993 0.993 1
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 49 1 | b = Iris-versicolor
0 0 50 | c = Iris-virginica

D. Plot RoC Curves

Procedure:

1. Load the dataset (Iris-2D. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under bayes section.
3. In which we selected Naïve-Bayes algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix.
5. For plotting RoC Curves, we need to right click on ―bayes.NaiveBayes‖ for getting more options, In which we will select the ―Visualize Threshold Curve‖ & go to any class (Iris-setosa, Iris-versicolor, Iris-Virgincia) as shown in below snapshot.
6. After selecting an class, RoC (Receiver Operating Characteristic) Curve plot will be displayed which has X-Axis –False Positive (FP) rate and Y-Axis – True Positive (TP) rate.

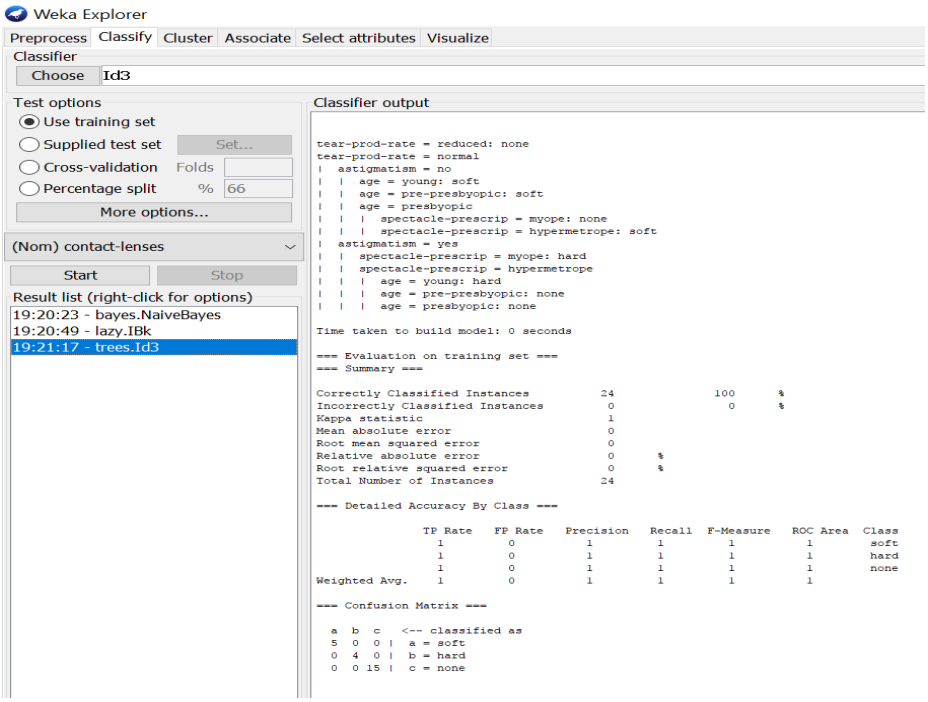

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
20
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
18:31:28 - bayes.NaiveBayes
Classifier output
tear-prod-rate = reduced: none
tear-prod-rate = normal
| astigmatism = no
| | age = young: soft
| | age = pre-presbyopic: soft
| | age = presbyopic
| | | spectacle-prescrip = myope: none
| | | spectacle-prescrip = hypermetrope: soft
| astigmatism = yes
| | spectacle-prescrip = myope: hard
| | spectacle-prescrip = hypermetrope
| | | age = young: hard
| | | age = pre-presbyopic: none
| | | age = presbyopic: none
Precision Recall F-Measure ROC Area Class
soft
hard
none
a = soft
b = hard
c = none
View in main window
View in separate window
Save result buffer
Delete result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Iris-setosa
Iris-versicolor
Iris-virginica


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
20
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
18:31:28 - bayes.NaiveBayes
Classifier output
Weka Classifier Visualize: ThresholdCurve. (Class value Iris-versicolor)
X: False Positive Rate (Num)
Y: True Positive Rate (Num)
Colour: Threshold (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot (Area under ROC = 0.9976)
Class colour

E. Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

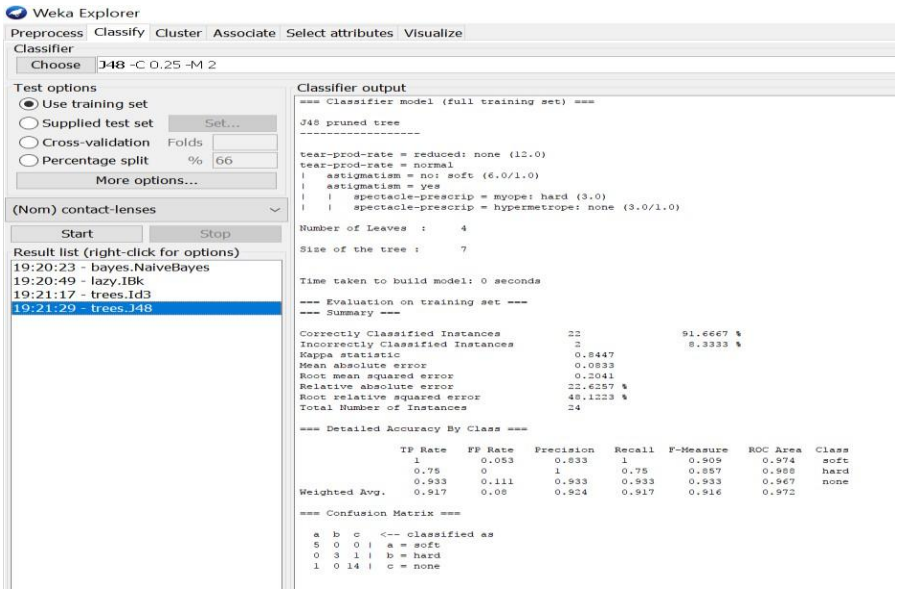
Procedure for ID3:

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under trees section.
3. In which we selected ID3 algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
Id3
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
Percentage split
%
66
More options...
(Nom) contact-lenses
Start
Stop
Result list (right-click for options)
19:20:23 - bayes.NaiveBayes
19:20:49 - lazy.IBk
19:21:17 - trees.Id3
Classifier output
tear-prod-rate = reduced: none
tear-prod-rate = normal
| astigmatism = no
| | age = young: soft
| | age = pre-presbyopic: soft
| | age = presbyopic
| | | spectacle-prescrip = myope: none
| | | spectacle-prescrip = hypermetrope: soft
| astigmatism = yes
| | spectacle-prescrip = myope: hard
| | spectacle-prescrip = hypermetrope
| | | age = young: hard
| | | age = pre-presbyopic: none
| | | age = presbyopic: none
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 24 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0
Root mean squared error 0
Relative absolute error 0 %
Root relative squared error 0 %
Total Number of Instances 24
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 soft
1 0 1 1 1 1 hard
1 0 1 1 1 1 none
Weighted Avg. 1 0 1 1 1 1
=== Confusion Matrix ===
a b c <-- classified as
5 0 0 | a = soft
0 4 0 | b = hard
0 0 15 | c = none

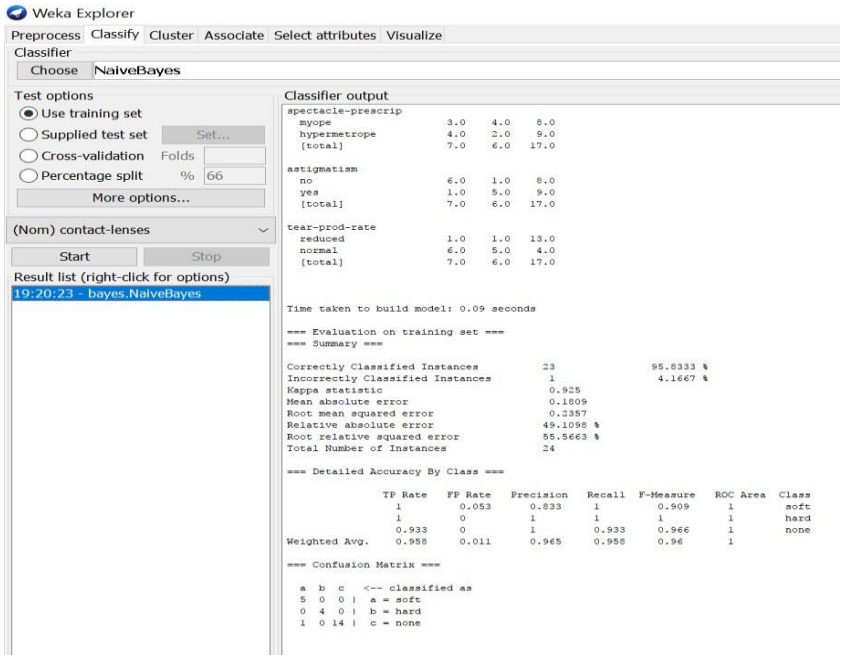
Procedure for J48:

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under trees section.
3. In which we selected J48 algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds
Percentage split % 66
More options...
(Nom) contact-lenses
Start Stop
Result list (right-click for options)
19:20:23 - bayes.NaiveBayes
19:20:49 - lazy.IBk
19:21:17 - trees.Id3
19:21:29 - trees.J48
Classifier output
=== Classifier model (full training set) ===
J48 pruned tree
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
| astigmatism = no: soft (6.0/1.0)
| astigmatism = yes
| | spectacle-prescrip = myope: hard (3.0)
| | spectacle-prescrip = hypermetrope: none (3.0/1.0)
Number of Leaves : 4
Size of the tree : 7
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 22 91.6667 %
Incorrectly Classified Instances 2 8.3333 %
Kappa statistic 0.8447
Mean absolute error 0.0833
Root mean squared error 0.2041
Relative absolute error 22.6257 %
Root relative squared error 48.1223 %
Total Number of Instances 24
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0.053 0.833 1 0.909 0.974 soft
0.75 0 1 0.75 0.857 0.988 hard
0.933 0.111 0.933 0.933 0.933 0.967 none
Weighted Avg. 0.917 0.08 0.924 0.917 0.916 0.972
=== Confusion Matrix ===
a b c <-- classified as
5 0 0 | a = soft
0 3 1 | b = hard
1 0 14 | c = none

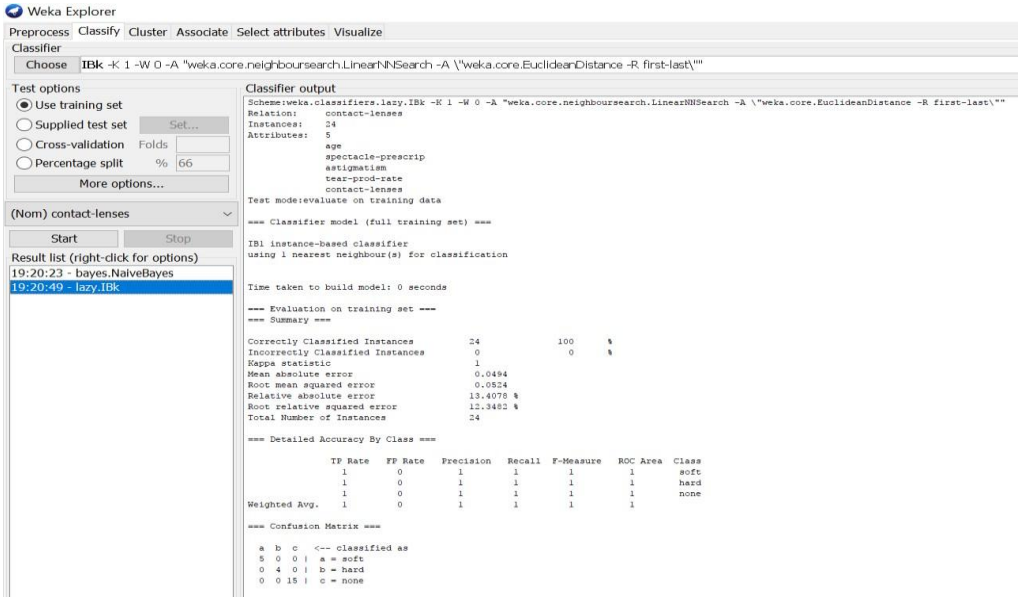
Procedure for Naïve-Bayes:

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassification algorithms under bayes section.
3. In which we selected Naïve-Bayes algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose NaiveBayes
Test options
Use training set
Supplied test set Set...
Cross-validation Folds
Percentage split % 66
More options...
(Nom) contact-lenses
Start Stop
Result list (right-click for options)
19:20:23 - bayes.NaiveBayes
Classifier output
spectacle-prescrip
myope 3.0 4.0 8.0
hypermetrope 4.0 2.0 9.0
[total] 7.0 6.0 17.0
astigmatism
no 6.0 1.0 8.0
yes 1.0 5.0 9.0
[total] 7.0 6.0 17.0
tear-prod-rate
reduced 1.0 1.0 13.0
normal 6.0 5.0 4.0
[total] 7.0 6.0 17.0
Time taken to build model: 0.09 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 23 95.8333 %
Incorrectly Classified Instances 1 4.1667 %
Kappa statistic 0.925
Mean absolute error 0.1809
Root mean squared error 0.2357
Relative absolute error 49.1098 %
Root relative squared error 55.5663 %
Total Number of Instances 24
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0.053 0.833 1 0.909 1 soft
1 0 1 1 1 1 hard
0.933 0 1 0.933 0.966 1 none
Weighted Avg. 0.958 0.011 0.965 0.958 0.96 1
=== Confusion Matrix ===
a b c <-- classified as
5 0 0 | a = soft
0 4 0 | b = hard
1 0 14 | c = none

Procedure for K-Nearest Neighbour (IBK):

1. Load the dataset (Contact-Lenses. arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclassifcation algorithms under lazy section.
3. In which we selected K-Nearest Neighbour (IBK) algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get detailed accuracy by class consists of F-measure, TP rate, FP rate, Precision, Recall values& Confusion Matrix as represented below.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set Set...
Cross-validation Folds
Percentage split % 66
More options...
(Nom) contact-lenses
Start Stop
Result list (right-click for options)
19:20:23 - bayes.NaiveBayes
19:20:49 - lazy.IBk
Classifier output
Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation: contact-lenses
Instances: 24
Attributes: 5
age
spectacle-prescrip
astigmatism
tear-prod-rate
contact-lenses
Test mode:evaluate on training data
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 24 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0.0494
Root mean squared error 0.0524
Relative absolute error 13.4078 %
Root relative squared error 12.3482 %
Total Number of Instances 24
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 soft
1 0 1 1 1 1 hard
1 0 1 1 1 1 none
Weighted Avg. 1 0 1 1 1 1
=== Confusion Matrix ===
a b c <-- classified as
5 0 0 | a = soft
0 4 0 | b = hard
0 0 15 | c = none

By observing all these algorithms (ID3, K-NN, J48 & Naïve Bayes) results, we will conclude that

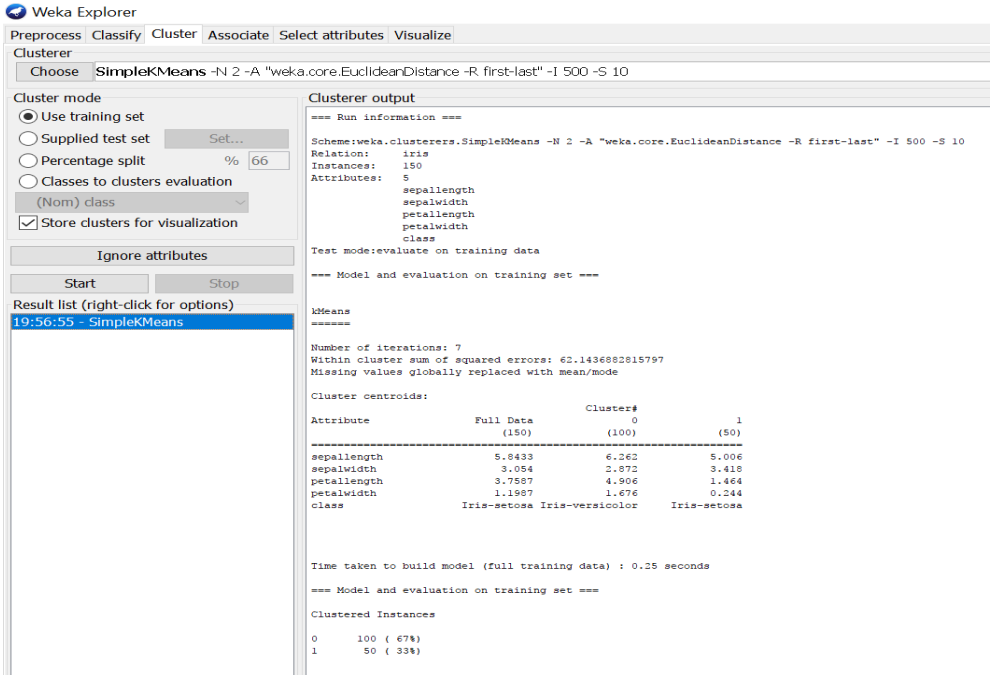
Hence,
ID3 Algorithm‟saccuracy & performance is best.
J48 Algorithm‟s accuracy & performance is poor.

Experiment 4
Demonstrate performing clustering on data sets

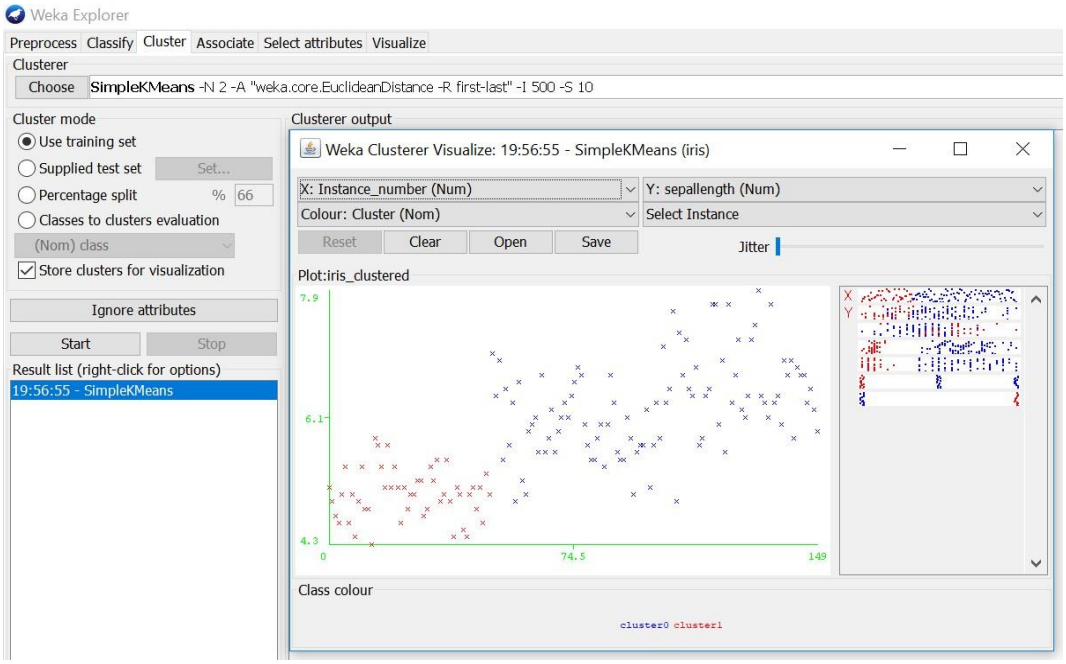
F. Load each dataset into Weka and run simple k-means clustering algorithm with different values of k (number of desired clusters). Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.

Procedure:

1. Load the dataset (Iris.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see differentclustering algorithms under lazy section.
3. In which we selected Simple K-Means algorithm & click on start option with ―use training set‖ test option enabled.
4. Then we will get the sum of squared errors, centroids, No. of iterations & clustered instances as represented below.

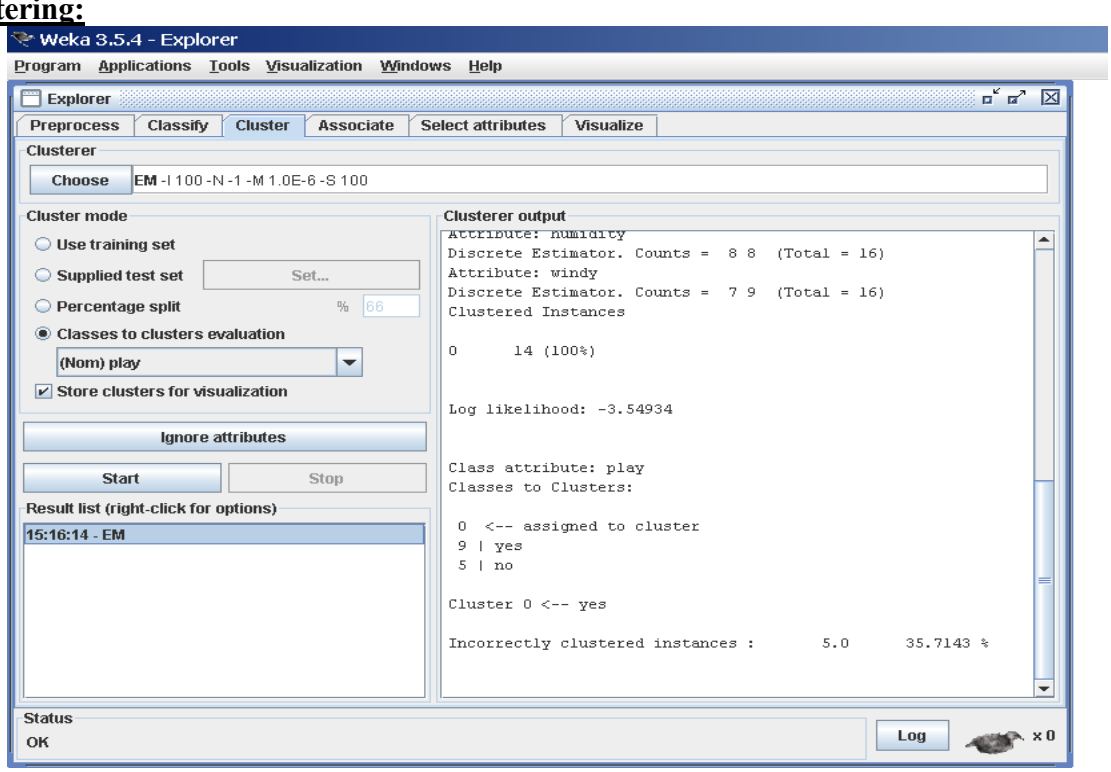


5. If we right click on simple k means, we will get more options in which ―Visualize cluster assignments‖ should be selected for getting cluster visualization as shown below.



G. Explore other clustering techniques available in Weka.

Clustering:



Selecting a Clusterer:

By now you will be familiar with the process of selecting and configuring objects.Clicking on the clustering scheme listed in the Clusterer box at the top of thewindow brings up aGenericObjectEditor dialog with which to choose a newclustering scheme.

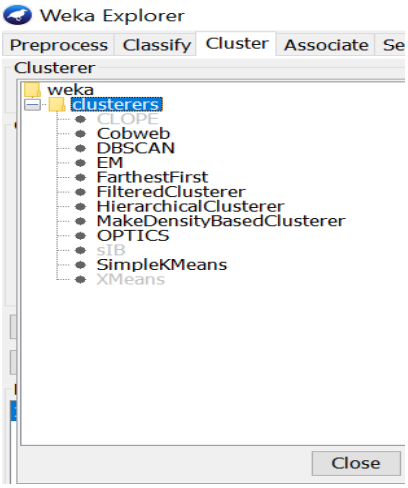
Cluster Modes:

The Cluster mode box is used to choose what to cluster and how to evaluatethe results. The first three options are the same as for classification: Use training set, supplied test set and Percentage split, now the data is assigned to clusters instead of trying to predict a specific class.The fourth mode, Classes to clusters evaluation, compares how well thechosen clusters match up with a pre-assigned class in the data. The drop-downbox below this option selects the class, just as in the Classify panel.An additional option in the Cluster mode box, the Store clusters forvisualization tick box, determines whether or not it will be possible to visualizethe clusters once training is complete. When dealing with datasets that are solarge that memory becomes aproblem it may be helpful to disable this option.

Ignoring Attributes:

Often, some attributes in the data should be ignored when clustering. TheIgnore attributes button brings up a small window that allows you to selectwhich attributes are ignored. Clicking on an attribute in the window highlightsit, holding down the SHIFT key selects a range of consecutive attributes, andholding down CTRL toggles individual attributes on and off. To cancel theselection, back out with the Cancel button. To activate it, click the Selectbutton. The next time clustering is invoked, the selected attributes are ignored.

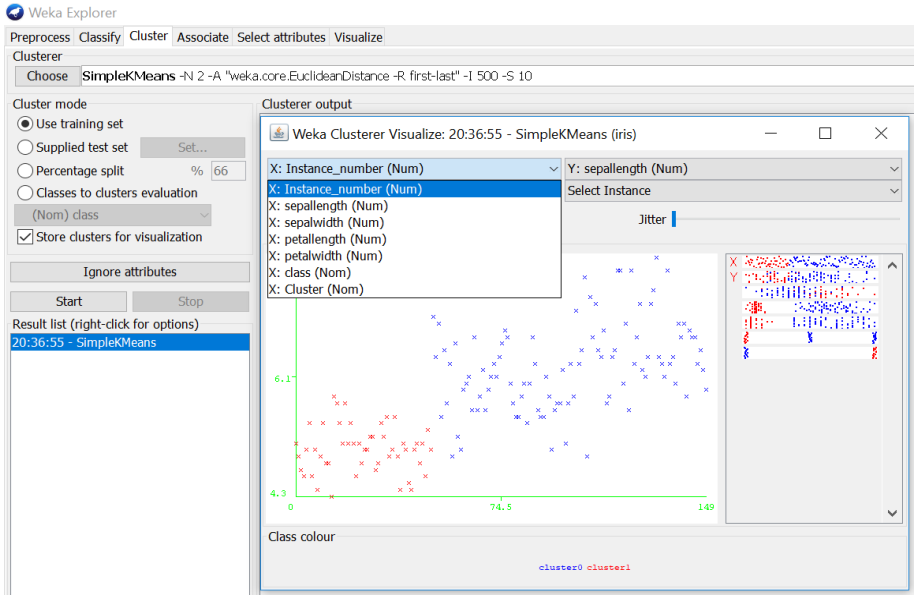
There are 12 clustering algorithms available in weka tool. They are shown below.



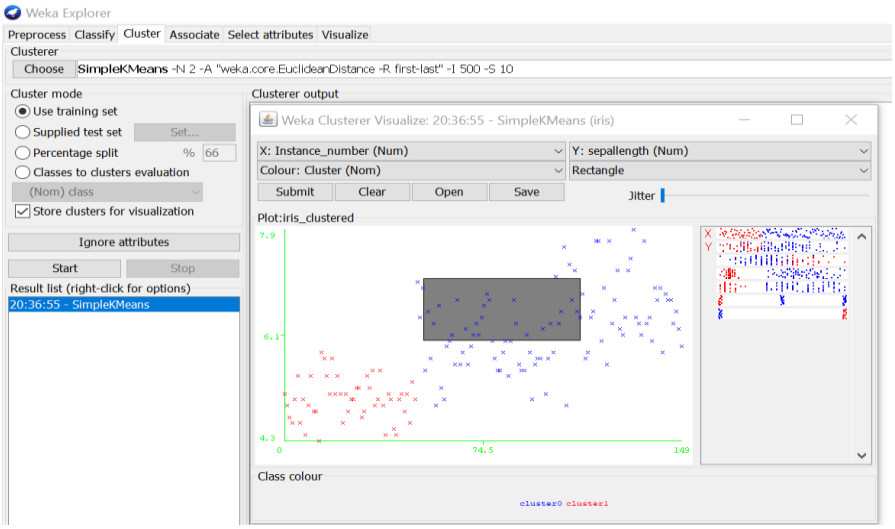
Through visualize cluster assignments, we can clearly see the clusters in graphical visualization.

H. Explore visualization features of Weka to visualize the clusters. Derive interesting insights and explain.

- If we right click on simple k means, we will get more options in which ―Visualize cluster assignments‖ should be selected for getting cluster visualization as shown below.
- In that cluster visualization we are having different features to explore by changing the X-axis, Y-axis, Color, Jitter& Select instance (Rectangle, Polygon & Polyline) for getting different sets of cluster outputs.



- As shown in above screenshot, all the dataset (Iris.arff) tuples are represented in X-axis & in similar way it will represented for y-axis also. For each cluster, the color will be different. In the above figure, there are two clusters which are represented in blue & red colors.
- In the select instance we can select different shapes for choosing clustered area as shown in below screenshot, rectangle shape is selected.



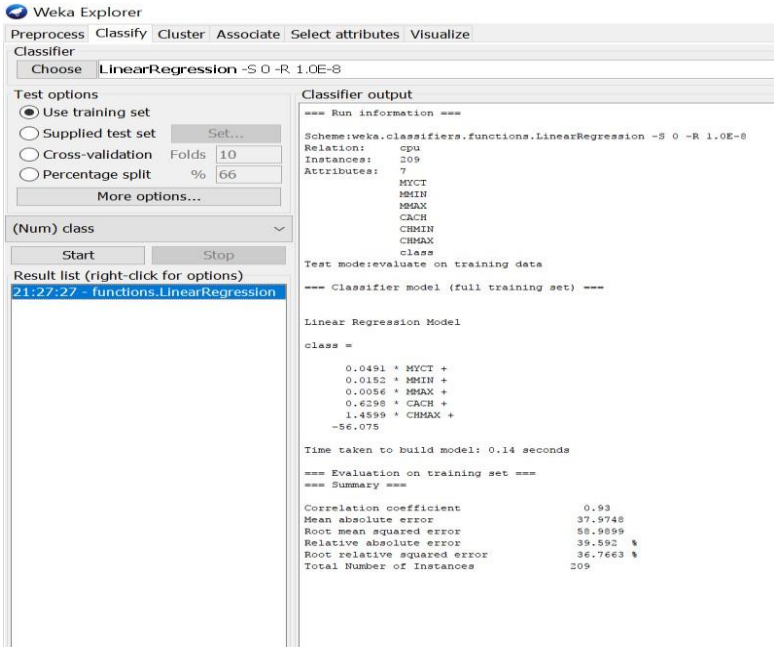
- By this visualization feature we can observe different clustering outputs for an dataset by changing those X-axis, Y-axis, Color & Jitter options.

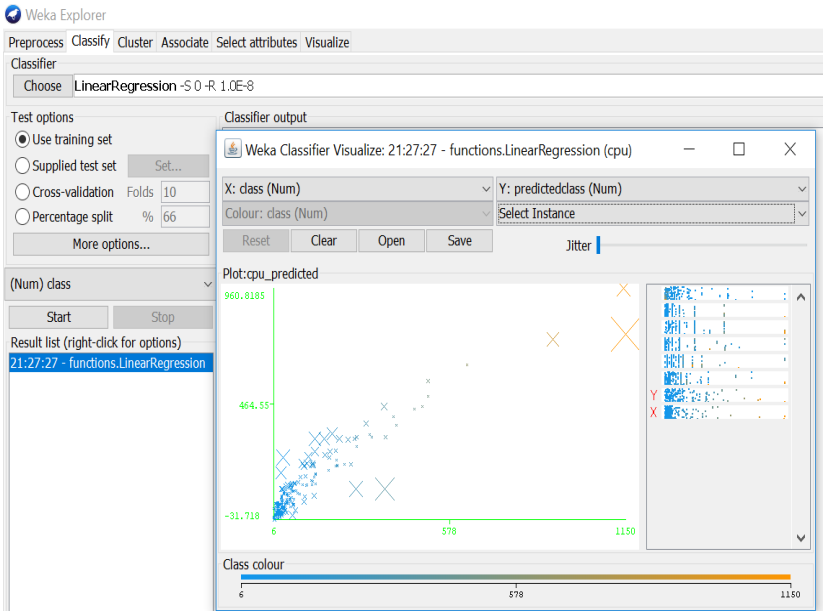
4. Demonstrate performing Regression on data sets

A. Load each dataset into Weka and build Linear Regression model. Study the clusters formed. Use Training set option. Interpret the regression model and derive patterns and conclusions from the regression results.

Procedure:

1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Linear Regression algorithm & click on start option with use training set option.
4. Then we will get regression model & its result as shown below.
5. The patterns are visually mentioned below for regression model through visualize classifier errors option which is available in right click options.

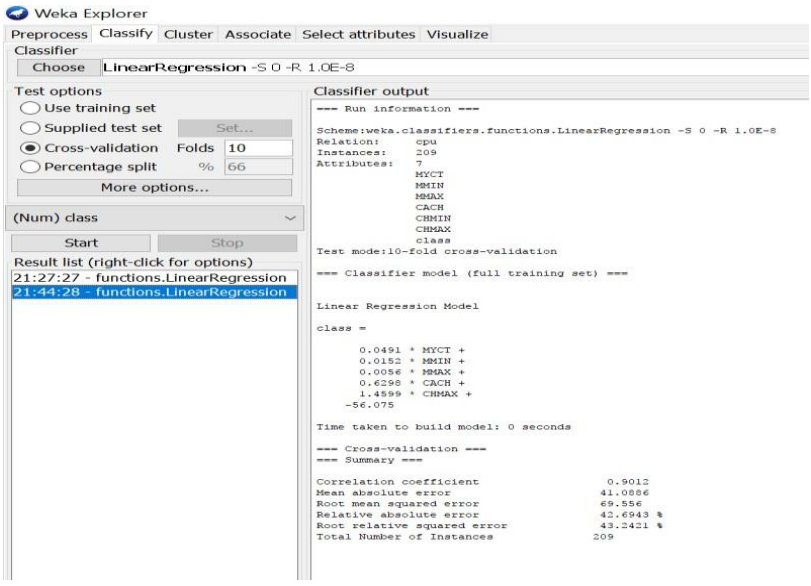




B. Use options cross-validation and percentage split and repeat running the Linear Regression Model. Observe the results and derive meaningful results.

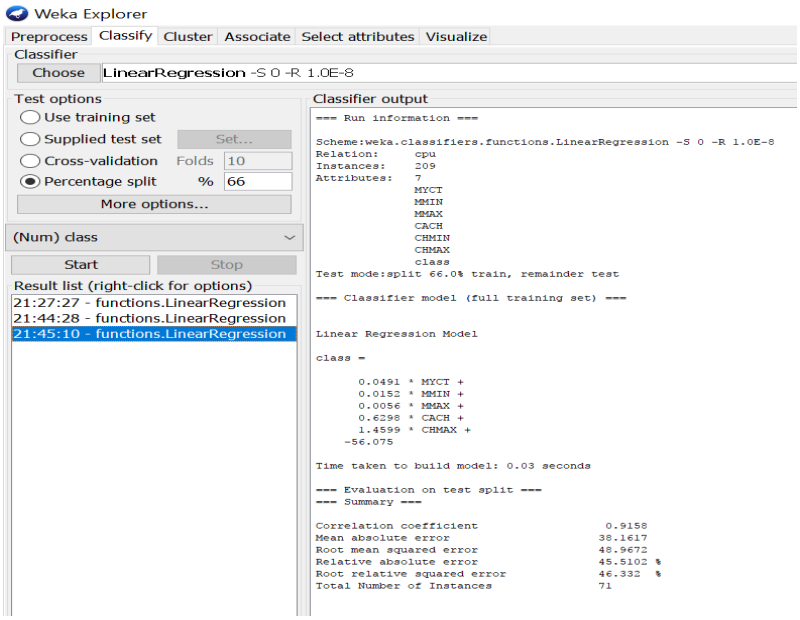
Procedure for cross-validation:

1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Linear Regression algorithm & click on start option with cross validation option with 10 folds.
4. Then we will get regression model & its result as shown below.



Procedure for percentage split:

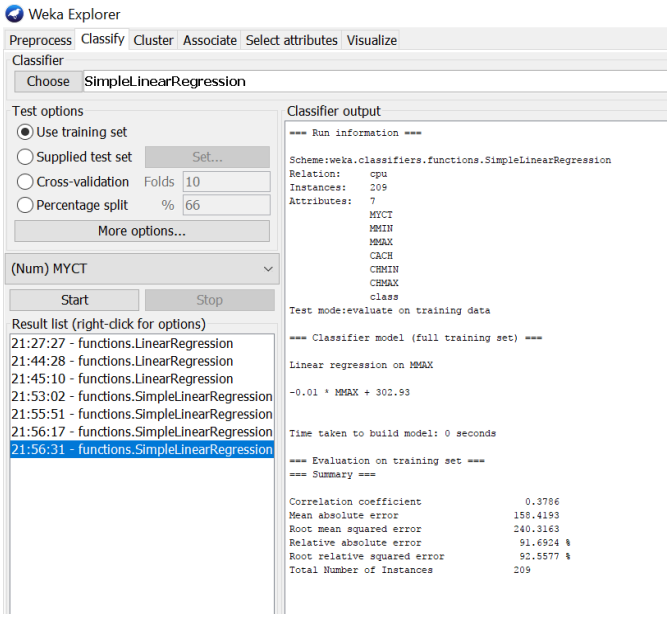
1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Linear Regression algorithm & click on start option with percentage split option with 66% split.
4. Then we will get regression model & its result as shown below.



C. Explore Simple linear regression technique that only looks at one variable

Procedure:

1. Load the dataset (Cpu.arff) into weka tool
2. Go to classify option & in left-hand navigation bar we can see different classification algorithms under functions section.
3. In which we selected Simple Linear Regression algorithm & click on start option with use training set option with one variable (MYCT).
4. Then we will get regression model & its result as shown below.


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
SimpleLinearRegression
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Num) MYCT
Start
Stop
Result list (right-click for options)
21:27:27 - functions.LinearRegression
21:44:28 - functions.LinearRegression
21:45:10 - functions.LinearRegression
21:53:02 - functions.SimpleLinearRegression
21:55:51 - functions.SimpleLinearRegression
21:56:17 - functions.SimpleLinearRegression
21:56:31 - functions.SimpleLinearRegression
Classifier output
=== Run information ===
Scheme:weka.classifiers.functions.SimpleLinearRegression
Relation: cpu
Instances: 209
Attributes: 7
MYCT
MMIN
MMAX
CACH
CHMIN
CHMAX
class
Test mode:evaluate on training data
=== Classifier model (full training set) ===
Linear regression on MMAX
-0.01 * MMAX + 302.93
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correlation coefficient 0.3786
Mean absolute error 158.4193
Root mean squared error 240.3163
Relative absolute error 91.6924 %
Root relative squared error 92.5577 %
Total Number of Instances 209

5. Demonstrate knowledge flow application on data sets into Weka/R

• Develop a knowledge flow layout for finding strong association rules by using Apriori, FP Growth algorithms

• Set up the knowledge flow to load an ARFF (batch mode) and perform a cross validation using J48 algorithm

• Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree

Frequent Pattern (FP) Growth Algorithm In Data Mining

FP Tree

Frequent Pattern Tree is a tree-like structure that is made with the initial itemsets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the itemset.

The root node represents null while the lower nodes represent the itemsets. The association of the nodes with the lower nodes that is the itemsets with the other itemsets are maintained while forming the tree.

Frequent Pattern Algorithm Steps

The frequent pattern growth method lets us find the frequent pattern without candidate generation.

Let us see the steps followed to mine the frequent pattern using frequent pattern growth algorithm:

#1) The first step is to scan the database to find the occurrences of the itemsets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.

#2) The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.

#3) The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The itemset with the max count is taken at the top, the next itemset with lower count and so on. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.

#4) The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch (for example in the 1st transaction), then this transaction branch would share a common prefix to the root.

This means that the common itemset is linked to the new node of another itemset in this transaction.

#5) Also, the count of the itemset is incremented as it occurs in the transactions. Both the common node and new node count is increased by 1 as they are created and linked according to transactions.

#6) The next step is to mine the created FP Tree. For this, the lowest node is examined first along with the links of the lowest nodes. The lowest node represents the frequency pattern length 1. From this, traverse the path in the FP Tree. This path or paths are called a conditional pattern base. Conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

#7) Construct a Conditional FP Tree, which is formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.

#8) Frequent Patterns are generated from the Conditional FP Tree.

Apriori Algorithm

Transaction ID	Items Bought
T1	Bread, Butter, Milk
T2	Bread, Butter
T3	Bread, Milk
T4	Butter, Milk
T5	Bread, Milk

Step 1 : Setting the parameters

- **Minimum Support Threshold:** 50% (item must appear in at least 3/5 transactions). This threshold is formulated from this formula:

Support(A)=Number of transactions containing itemset ATotal number of transactionsSupport(A)=Total number of transactionsNumber of transactions containing itemset A

- **Minimum Confidence Threshold:** 70% (You can change the value of parameters as per the use case and problem statement). This threshold is formulated from this formula:

Confidence(X→Y)=Support(X∪Y)Support(X)Confidence($X\rightarrow Y$)=Support(X)Support($X\cup Y$)

Step 2: Find Frequent 1-Itemsets

Lets count how many transactions include each item in the dataset (calculating the frequency of each item).

Item	Support Count	Support %
Bread	4	80%
Butter	3	60%
Milk	4	80%

Frequent 1-Itemsets

All items have support% ≥ 50%, so they qualify as frequent 1-itemsets. if any item has support% < 50%, It will be omitted out from the frequent 1- itemsets.

Step 3: Generate Candidate 2-Itemsets

Combine the frequent 1-itemsets into pairs and calculate their support. For this use case we will get 3 item pairs (bread,butter) , (bread,ilk) and (butter,milk) and will calculate the support similar to step 2

Item Pair	Support Count	Support %
Bread, Butter	2	40%
Bread, Milk	3	60%
Butter, Milk	2	40%

Candidate 2-Itemsets

Frequent 2-itemsets: {Bread, Milk} meet the 50% threshold but {Butter, Milk} and {Bread ,Butter} doesn't meet the threshold, so will be committed out.

Step 4: Generate Candidate 3-Itemsets

Combine the frequent 2-itemsets into groups of 3 and calculate their support. for the triplet we have only got one case i.e {bread,butter,milk} and we will calculate the support.

Item Triplet	Support Count	Support %
Bread, Butter, Milk	1	40%

Candidate 3-Itemsets

Since this does not meet the 50% threshold, there are no frequent 3-itemsets.

Step 5: Generate Association Rules

Now we generate rules from the frequent itemsets and calculate confidence.

Rule 1: If Bread → Butter (if customer buys bread, the customer will buy butter also)

- Support of {Bread, Butter} = 2.
- Support of {Bread} = 4.

- Confidence = 2/4 = 50% (Failed threshold).

Rule 2: If Butter → Bread (if customer buys butter, the customer will buy bread also)

- Support of {Bread, Butter} = 3.
- Support of {Butter} = 3.
- Confidence = 3/3 = 100% (Passes threshold).

Rule 3: If Bread → Milk (if customer buys bread, the customer will buy milk also)

- Support of {Bread, Milk} = 3.
- Support of {Bread} = 4.
- Confidence = 3/4 = 75% (Passes threshold).

The Apriori Algorithm, as demonstrated in the bread-butter example, is widely used in modern startups like Zomato, Swiggy and other food delivery platforms. These companies use it to perform market basket analysis which helps them identify customer behaviour patterns and optimise recommendations.

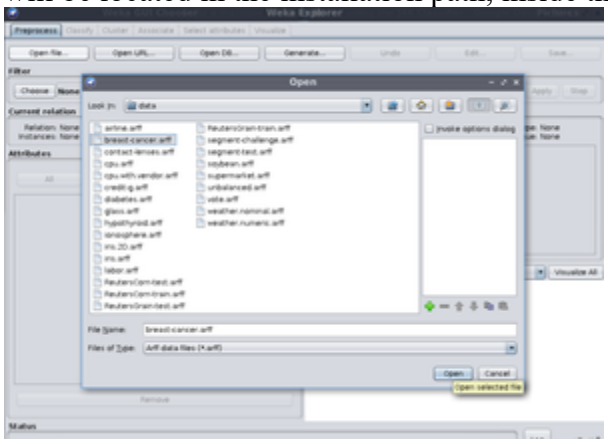
Building a Machine Learning Model Using J48 Classifier

Steps to follow:

Step 1: Create a model using GUI

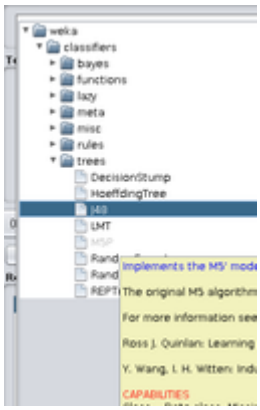
Step 2: After opening Weka click on the "Explorer" Tab

Step 3: In the "Preprocess" Tab Click on "Open File" and select the "breast-cancer.arff" file which will be located in the installation path, inside the data folder.

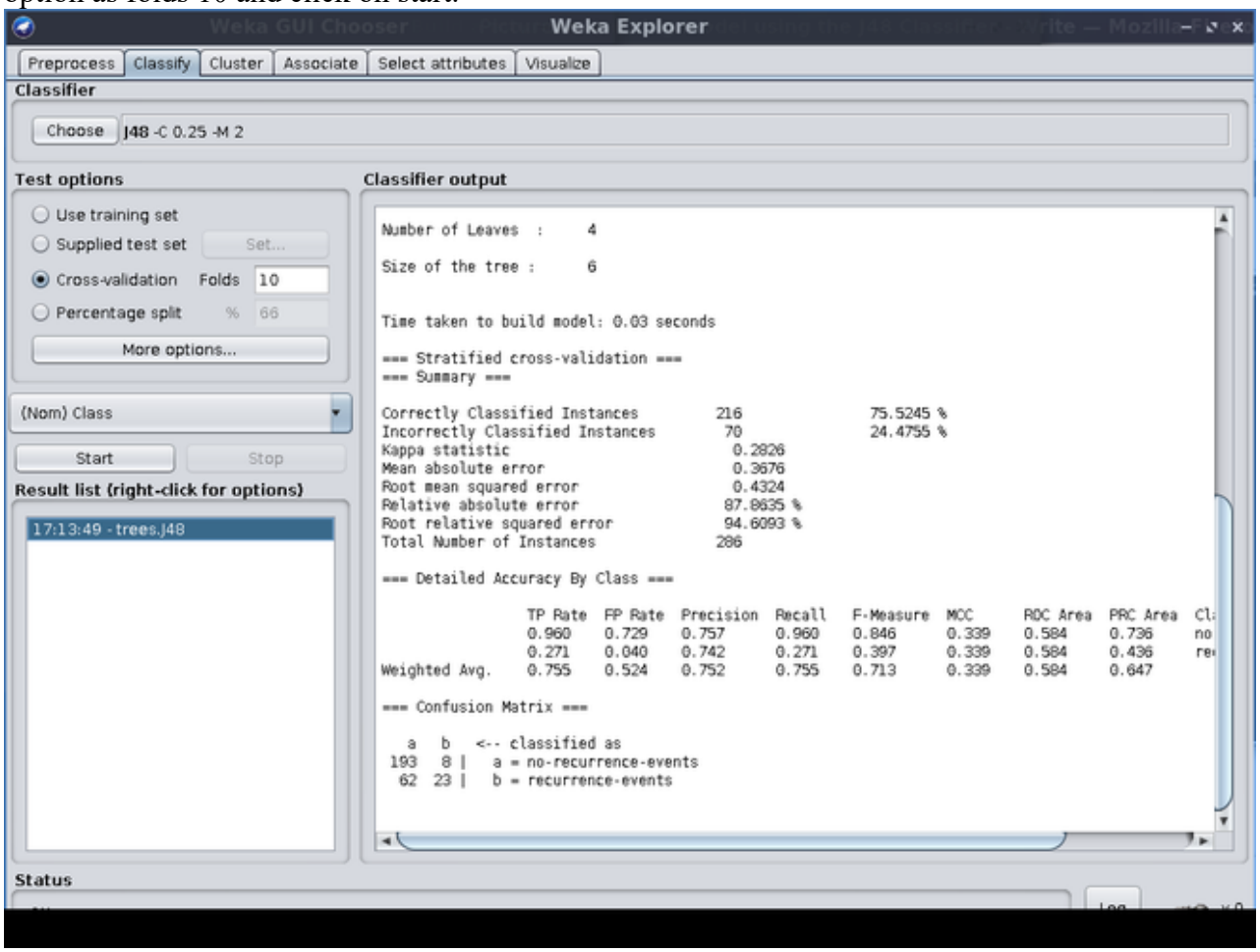


In this tab, you can view all the attributes and play with them.

Step 4: In the "Classify" tab click on the choose button. Now under weka/classifiers/trees/ select J48



Step 5: Now one can click on the J48 Classifier selection and play around with it like changing batch size, confidence factor, etc. There under "Test Options" we'll use the default cross-validation option as folds 10 and click on start.



Implementation:

Now we are done with discussing that Weka has Java API that you can use to create machine learning models so di now let us create a model using API

Example

// Java Program for Creating a Model Based on J48 Classifier

// Importing required classes

import java.io.BufferedReader;

```
import java.io.FileReader;
import java.util.Random;
import weka.classifiers.Evaluation;
import weka.classifiers.trees.J48;
import weka.core.Instances;

// Main class
public class BreastCancer {

    // Main driver method
    public static void main(String args[])
    {

        // Try block to check for exceptions
        try {

            // Creating J48 classifier
            J48 j48Classifier = new J48();

            // Dataset path
            String breastCancerDataset
                = "/home/droid/Tools/weka-3-8-5/data/breast-cancer.arff";

            // Create bufferedreader to read the dataset
            BufferedReader bufferedReader
                = new BufferedReader(
                    new FileReader(breastCancerDataset));

            // Create dataset instances
            Instances datasetInstances
                = new Instances(bufferedReader);

            // Set Target Class
            datasetInstances.setClassIndex(
                datasetInstances.numAttributes() - 1);

            // Evaluation
            Evaluation evaluation
                = new Evaluation(datasetInstances);

            // Cross Validate Model with 10 folds
            evaluation.crossValidateModel(
                j48Classifier, datasetInstances, 10,
                new Random(1));
            System.out.println(evaluation.toSummaryString(
                "\nResults", false));
        }
```

```
        // Catch block to check for rexceptions
        catch (Exception e) {

            // Print and display the display message
            // using getMessage() method
            System.out.println("Error Occurred!!!! \n"
                               + e.getMessage());
        }

        // Display message to be printed ion console
        // when program is successfully executed
        System.out.print("Successfully executed.");
    }
}
```

Output:

Successfully executed.

Demonstrate plotting multiple ROC curves in the same plot window by using j48 and Random forest tree

R-CODE: Train the models

```
#Open libraries
library(MASS)
library(caret)
library(e1071)
library(klaR)
library(gbm)
library(earth)
library(kernlab)
library(rpart)
library(randomForest)
library(mlbench)
library(adabag)
library(ada)
library(fastAdaboost)
library(xgboost)
library(C50)

##Produce a new version of the dataframe 'Clusters_Dummy' with the rows shuffled
NewClusters=Cluster_Dummy_2[sample(1:nrow(Cluster_Dummy_2)),]

#Produce a dataframe
NewCluster<-as.data.frame(NewClusters)

#display
print(NewCluster)

#Check the structure of the data
```

```
str(NewCluster)

#Number of rows
nrow(NewCluster)

#Split the data frame into 70% to 30% train and test data
training.parameters <- Cluster_Dummy_2$Country %>%
createDataPartition(p = 0.7, list = FALSE)
train.data <- NewClusters[training.parameters, ]
test.data <- NewClusters[-training.parameters, ]
sapply(train.data, summary) # Look at a summary of the training data

#########################################################
#FIT MODELS: Auxiliary function to train the models
#########################################################

fitControl <- trainControl(## 10-fold CV
                    method = "repeatedcv",
                    number = 10,
                    ## repeated ten times
                    repeats = 10,
                    classProbs = TRUE,
                    verbose = TRUE)


tuneLength <- 10
metric <- "Accuracy"

#####
#LDA#
#####
#Train the model
lda.fit.CV = train(Country ~ ., data=train.data, method="lda",
                trControl = fitControl, metric=metric, tuneLength = tuneLength)

lda.fit.CV

#######################################
# Stochastic Boosted Gradient Trees #
#######################################

gbmGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                    n.trees = (1:30)*50,
                    shrinkage = 0.1,
                    n.minobsinnode = 20)

#Stochastic Boosted Gradient Tree: model 1
gbmFit1 <- train(Country ~ ., data=train.data,   method = "gbm",  metric=metric,
```

```
            trControl = fitControl, tuneLength = tuneLength, verbose=FALSE)

gbmFit1

#Stochastic Boosted Gradient Tree: model 2
gbmFit2 <- train(Country ~ ., data=train.data, method = "gbm",  trControl = fitControl,
            metric=metric, tuneLength = tuneLength,  tuneGrid = gbmGrid, verbose=FALSE)

gbmFit2

#############################################################
# Multivariate Adaptive Regression Splines (MARS) model
#############################################################

# Step 1: Define the tuneGrid
marsGrid <-  expand.grid(nprune = c(2, 4, 6, 8, 10),
                  degree = c(1, 2, 3))

# Train the model using randomForest and predict on the training data itself.
model_mars = train(Country ~ .,  data=train.data,  method='earth',  metric='ROC', tuneGrid =
marsGrid, trControl = fitControl, tuneLength = tuneLength)

model_mars

#################################
# Single Vector Machine (SVM) #
#################################

model_svmRadial = train(Country ~ ., data=train.data, method='svmRadial',, metric=metric,
trControl = fitControl, tuneLength= tuneLength)

model_svmRadial

################################################
# Recursive Partitioning Classification Trees #
################################################

f <- as.formula(paste0("Country ~ ", paste0(names(train.data)[2:10], collapse = "+")))

rpart.ctrl <- rpart.control(minsplit = 5, minbucket = 5, cp = seq(0, 0.02, 0.0001))

dt.rpart <- train(form = f, data = train.data, method = "rpart", metric = metric, trControl =
fitControl, tuneGrid = rpart.ctrl, tuneLength= tuneLength)
dt.rpart

###############
Naive Bayes
#############
```

```
#Tune the model
nb_tune <- data.frame(usekernel = TRUE, fL = 0, adjust=seq(0, 5, by = 1))

#Train the model
model.nb = train(Country ~., data=train.data,'nb', trControl=fitControl, metric=metric,
tuneLength=tuneLength, tuneGrid = nb_tune, laplace = 0:3)
model.nb

##################
# Random Forest #
##################

model_rf = train(Country ~., data=train.data, method='rf', metric=metric, tuneLength= tuneLength,
trControl = fitControl)

model_rf

#######################################
# Regularized Discriminant Analysis #
#######################################

rdaGrid=data.frame(gamma = (0.4)/4, lambda = 3/4)

rdaFit <- train(Country ~ ., data =train.data, method = "rda", trControl = fitControl, tuneLength =
tuneLength, metric = "ROC")
rdaFit

#######################################
# Classification with Decision Tree #
#######################################

#Train the model
Decision_Fit <- train(Country ~ ., data =train.data, method = "C5.0", trControl = fitControl,
tuneLength = tuneLength, metric = "ROC")
Decision_Fit

###################################
# K-nearest neighbour classifier #
###################################

#Train the model
model_knn = caret::train(Country ~ ., data=train.data, method='knn',
                         tuneLength = tuneLength, metric=metric,
                         trControl = fitControl,
                         tuneGrid = expand.grid(k = seq(1, 101, by = 2)))
model_knn
```

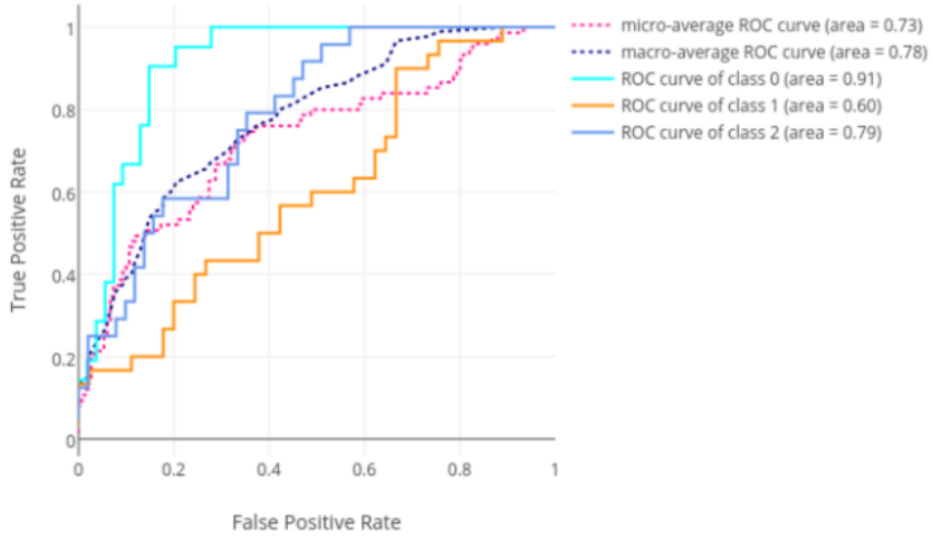
```
##############################
# Neural Network Classifier #
##############################

Neural_Fit <- train(Country ~ ., data =train.data, method = "nnet", trControl = fitControl,
tuneLength = tuneLength, metric = "ROC")
Neural_Fit
```

Produce the mult.roc() ojbects

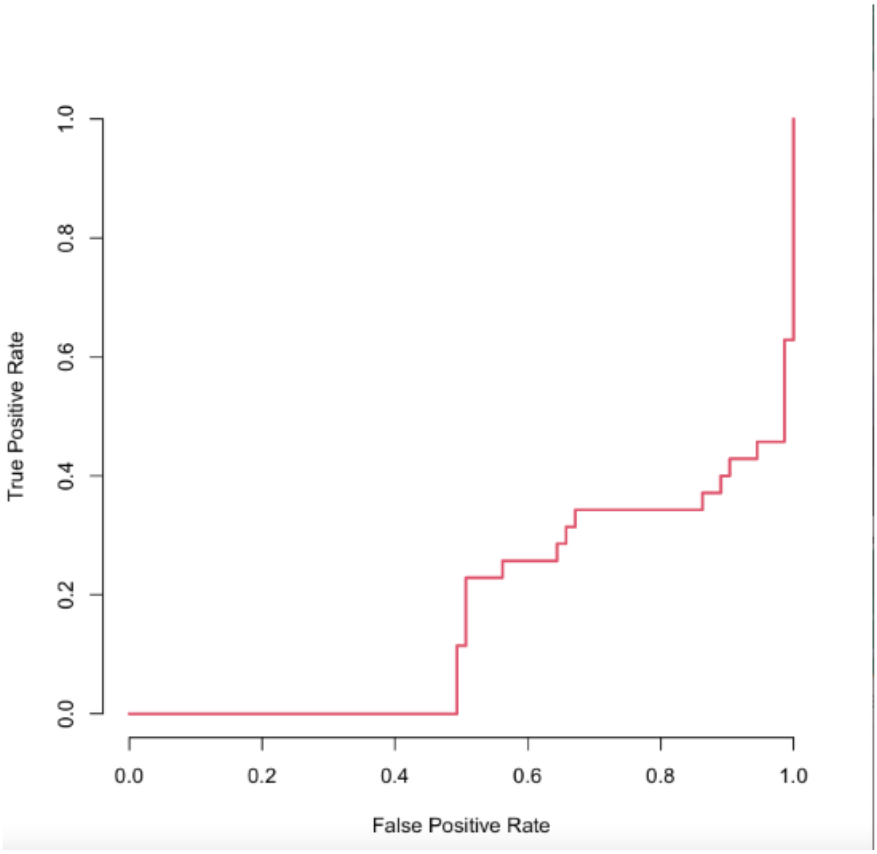
```
#LDA
roc_LDA <- multiclass.roc(test.data$Country, pred_LDA, levels=c("France", "Spain", "Holland"),
auc=TRUE)
roc_LDA

#Boosted Tree model 1
roc_Tree1 <- multiclass.roc(test.data$Country, pred_model_Tree1, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_Tree1

#Boosted Tree model 2
roc_Tree2 <- multiclass.roc(test.data$Country, pred_model_Tree2, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_Tree2

#Mars model
roc_Mars <- multiclass.roc(test.data$Country, Earth_fitted, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_Mars

#Single Vector Machine
roc_SVM <- multiclass.roc(test.data$Country, SVM_fitted, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_SVM

#Single Vector Machine
roc_RART <- multiclass.roc(test.data$Country, rpart_fitted, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_RART

#Naive Bayes
roc_nb <- multiclass.roc(test.data$Country, predict.nb$posterior, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_nb

#Single Vector Machine
roc_RART <- multiclass.roc(test.data$Country, rpart_fitted, levels=c("France", "Spain",
"Holland"))
roc_RART
```

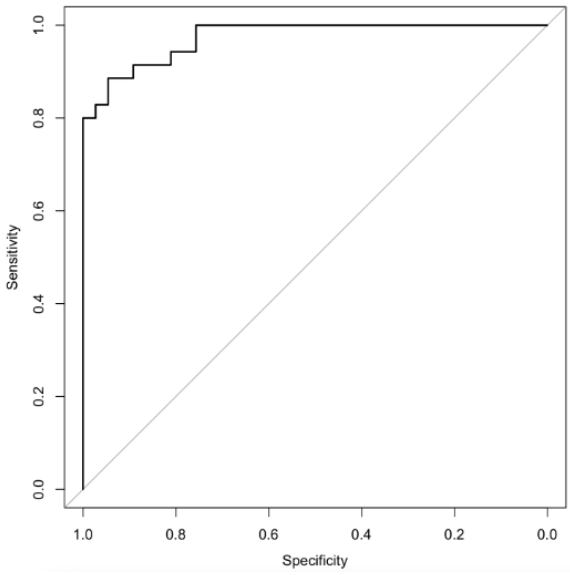
```
#Random Forest
roc_RF <- multiclass.roc(test.data$Country, pred_rf, levels=c("France", "Spain", "Holland"),
auc=TRUE)
roc_RF

#Regularized Discriminant Analysis
roc_RDA <- multiclass.roc(test.data$Country, pred_rda, levels=c("France", "Spain", "Holland"),
auc=TRUE)
roc_RDA

#Decision Trees
roc_Decision <- multiclass.roc(test.data$Country, pred_decision, levels=c("France", "Spain",
"Holland"), auc=TRUE)
roc_Decision

#K nearest neighbour
roc_knn <- multiclass.roc(test.data$Country, pred_knn, levels=c("France", "Spain", "Holland"),
auc=TRUE)
roc_knn

#Neural network
roc_net <- multiclass.roc(test.data$Country, pred_net, levels=c("France", "Spain", "Holland"), auc
= TRUE)
roc_net
```

Expected Output



Output

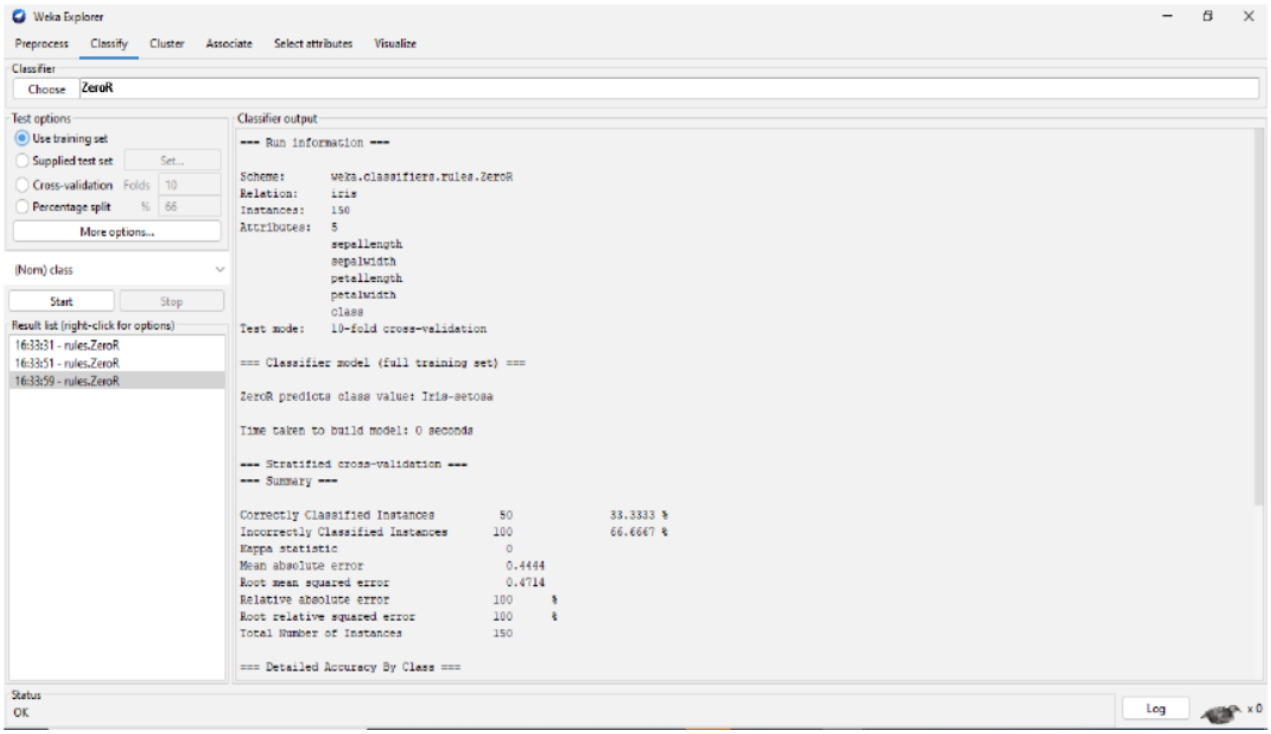

True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

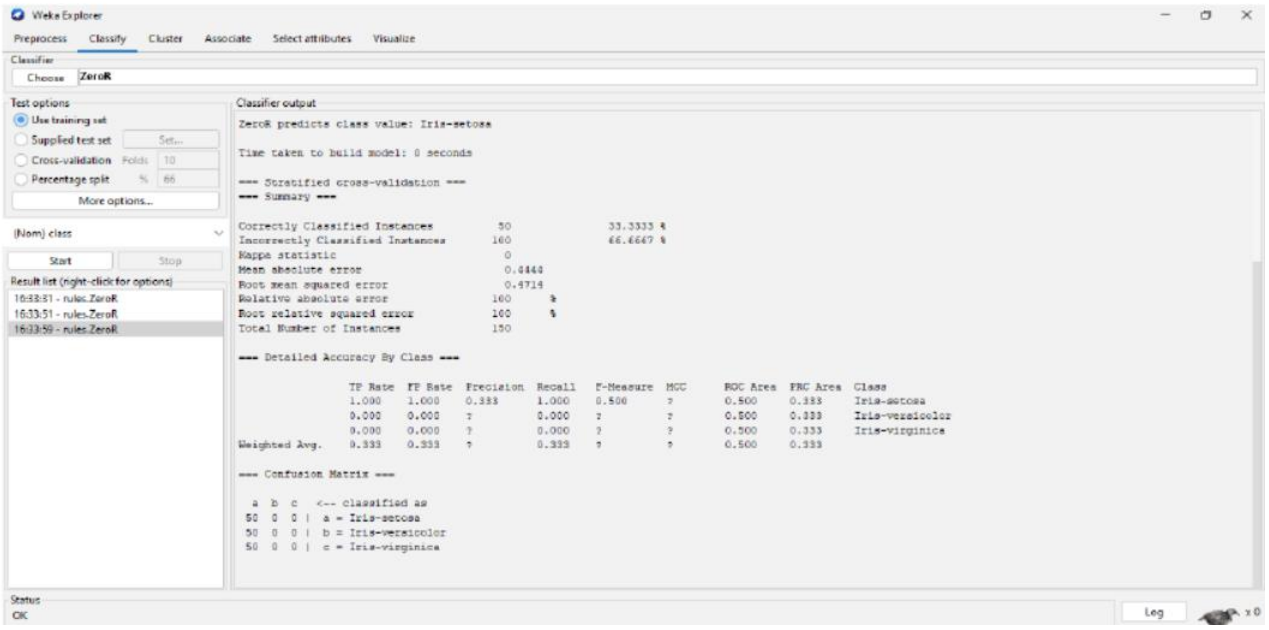

Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
Specificity

Experiment 7
Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observations.

ZeroR is the simplest classification method which relies on the target and ignores all predictors.

- ZeroR classifier simply predicts the majority category (class).
- Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
16:33:31 - rules.ZeroR
16:33:51 - rules.ZeroR
16:33:59 - rules.ZeroR
Classifier output
ZeroR predicts class value: Iris-setosa
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 50 33.3333 %
Incorrectly Classified Instances 100 66.6667 %
Kappa statistic 0
Mean absolute error 0.4444
Root mean squared error 0.4714
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 1.000 0.333 1.000 0.500 ? 0.500 0.333 Iris-setosa
0.000 0.000 ? 0.000 ? ? 0.500 0.333 Iris-versicolor
0.000 0.000 ? 0.000 ? ? 0.500 0.333 Iris-virginica
Weighted Avg. 0.333 0.333 ? 0.333 ? ? 0.500 0.333
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
50 0 0 | b = Iris-versicolor
50 0 0 | c = Iris-virginica
Status
OK
Log
x 0

Experiment 8
Write a java program to prepare a simulated data set with unique instances

Creating new dataset java
Dataset dt = new DefaultDataset (); // creation syntax for the dataset
for
{
(b=0, b<8, b++) // condition setting
Instnc inst_1 = Instnc.randomInstnc(12); // defining the instance for the dataset
Dt.add(inst_1);
}
//adding the instance for the dataset
This program is used for creating and iterating the entire dataset representing the car name and car characteristic when getting a sql query to be performed over it.

```
public
public
class Cars_dtset {
;
public String car_description;
public int car_no;
}
interface Actual_Query extends Bs_Query {
@Select("select car_name, car_description, car_no from Cars_dtset")
DataSet<Cars_dtset> getAllCars_dtset();
}
Actual_Query mq_0 = con.createQueryObject(Actual_Query.class);
DataSet rows = mq_0.getAllCars_dtset();
for (Cars_dtset mq_0: rows) {
System.out.println("CarName = " + mq_0.car_name);
System.out.println("CarDescription = " + mq_0.car_description);
}
```

Experiment-9
Write a Python program to generate frequent item sets / association rules using Apriori algorithm.

Program:

```
# installing the apyori package
!pip install apyori
# importing libraries
Import numpy as np
Import matplotlib.pyplot as plt
import pandas as pd
Data = pd.read_csv('/content/drive/MyDrive/Market_Basket_Optimisation.csv', header = None)
# Intializing the list
transacts = [ ]
# populating a list of transactions
for i in range(0, 7501):
transacts.append([str(Data.values[i,j]) for j in range(0, 20)])
fromapyori import apriori
rule = apriori(transactions = transacts, min_support = 0.003, min_confidence = 0.2, min_lift = 3,
min_length = 2, max_length = 2)
output = list(rule) # returns a non-tabular output
# putting output into a pandasdataframe
def inspect(output):
lhs   = [tuple(result[2][0][0])[0] for result in output]
rhs   = [tuple(result[2][0][1])[0] for result in output]
support    = [result[1] for result in output]
confidence = [result[2][0][2] for result in output]
lift       = [result[2][0][3] for result in output]
return list(zip(lhs, rhs, support, confidence, lift))
output_DataFrame = pd.DataFrame(inspect(results), columns = ['Left_Hand_Side',
'Right_Hand_Side', 'Support', 'Confidence', 'Lift'])
```

Output:

	Left_Hand_Side	Right_Hand_Side	Support	Confidence	Lift
0	light cream	chicken	0.004533	0.290598	4.843951
1	mushroom cream sauce	escalope	0.005733	0.300699	3.790833
2	pasta	escalope	0.005866	0.372881	4.700812
3	fromage blanc	honey	0.003333	0.245098	5.164271
4	herb & pepper	ground beef	0.015998	0.323450	3.291994
5	tomato sauce	ground beef	0.005333	0.377358	3.840659
6	light cream	olive oil	0.003200	0.205128	3.114710
7	whole wheat pasta	olive oil	0.007999	0.271493	4.122410
8	pasta	shrimp	0.005066	0.322034	4.506672

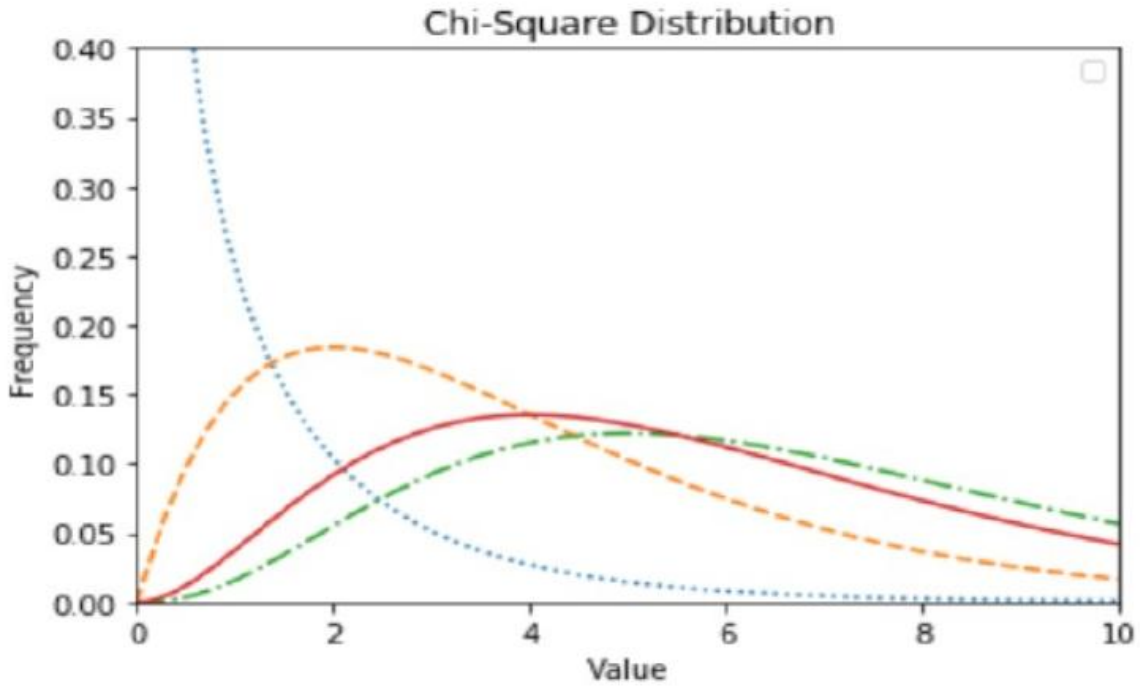
Experiment-10

Write a program to calculate chi-square value using Python. Report your observation.

Program:

```
fromscipy import stats
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 100)
fig,ax = plt.subplots(1,1)
linestyles = [':', '--', '-.', '-']
deg_of_freedom = [1, 4, 7, 6]
fordf, ls in zip(deg_of_freedom, linestyles):
ax.plot(x, stats.chi2.pdf(x, df), linestyle=ls)
plt.xlim(0, 10)
plt.ylim(0, 0.4)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Chi-Square Distribution')
plt.legend()
plt.show()
```

Output:



Experiment-11

Write a program of Naive Bayesian classification using Python programming language.

Program:

```
import numpy as np
genders = ["male", "female"]
persons = [ ]
with open("data/person_data.txt") as fh:
for line in fh:
persons.append(line.strip().split())
firstnames = {}
heights = {}
for gender in genders:
firstnames[gender] = [ x[0] for x in persons if x[4]==gender]
heights[gender] = [ x[2] for x in persons if x[4]==gender]
heights[gender] = np.array(heights[gender], np.int)
for gender in ("female", "male"):
print(gender + ":")
print(firstnames[gender][:10])
print(heights[gender][:10])
```

Output:

```
female:
['Stephanie', 'Cynthia', 'Katherine', 'Elizabeth', 'Carol', 'Christina', 'Beverly', 'Sharon', 'Denise',
'Rebecca']
[149 174 183 138 145 161 179 162 148 196]
male:
['Randy', 'Jessie', 'David', 'Stephen', 'Jerry', 'Billy', 'Earl', 'Todd', 'Martin', 'Kenneth']
[184 175 187 192 204 180 184 174 177 200]
```

Experiment-12
Implement a Java program to perform Apriori algorithm
Program:

```
import java.util.*;
import java.sql.*;
class Tuple
{
Set<Integer> itemset;
int support;
Tuple()
{
itemset = new HashSet<>();
support = -1;
}
Tuple(Set<Integer> s)
{
itemset = s;
support = -1;
}
Tuple(Set<Integer> s, int i)
{
itemset = s;
support = i;
}
}
class Apriori
{
static Set<Tuple> c;
static Set<Tuple> l;
static int d[][];
static int min_support;
public static void main(String args[]) throws Exception
{
getDatabase();
c = new HashSet<>();
l = new HashSet<>();
Scanner scan = new Scanner(System.in);
int i, j, m, n;
System.out.println("Enter the minimum support (as an integer value):");
min_support = scan.nextInt();
Set<Integer> candidate_set = new HashSet<>();
for(i=0 ; i < d.length ; i++)
{
System.out.println("Transaction Number: " + (i+1) + ":");
for(j=0 ; j < d[i].length ; j++)
{
System.out.print("Item number " + (j+1) + " = ");
System.out.println(d[i][j]);
```

```
candidate_set.add(d[i][j]);
}
}
Iterator<Integer> iterator = candidate_set.iterator();
while(iterator.hasNext())
{
Set<Integer> s = new HashSet<>();
s.add(iterator.next());

temp = new LinkedList<>();
}
temp.add(object);
m.put(list_no, temp);
}
Set<Integer> keyset = m.keySet();
d = new int[keyset.size()][ ];
Iterator<Integer> iterator = keyset.iterator();
int count = 0;
while(iterator.hasNext())
{
temp = m.get(iterator.next());
Integer[ ] int_arr = temp.toArray(new Integer[0]);
d[count] = new int[int_arr.length];
for(int i=0 ; i < d[count].length ; i++)
{
d[count][i] = int_arr[i].intValue();
}
count++;
}
}
}
```

Output:

```
Enter the minimum support (as a floating point value, 0<x<1):
0.5
Transaction Number: 1:
Item number 1 = 1
Item number 2 = 3
Item number 3 = 4
Transaction Number: 2:
Item number 1 = 2
Item number 2 = 3
Item number 3 = 5
Transaction Number: 3:
Item number 1 = 1
Item number 2 = 2
Item number 3 = 3
Item number 4 = 5
Transaction Number: 4:
```

```
Item number 1 = 2
Item number 2 = 5
-+- L -+-
[1] : 2
[3] : 3
[2] : 3
[5] : 3
-+- L -+-
[2, 3] : 2
[3, 5] : 2
[1, 3] : 2
[2, 5] : 3
-+- L -+-
[2, 3, 5] : 2
=+= FINAL LIST =+=
[2, 3, 5] : 2
```

Experiment 13
Write a R program to cluster your choice of data using simple k-means algorithm using JDK

```
# Example of k-means clustering in R

# Step 1: Generate or load your data
# Example: Generating random data
set.seed(123)  # For reproducibility
data <- matrix(rnorm(100), ncol=2)  # Example data with 100 samples and 2 variables

# Step 2: Apply k-means algorithm
k <- 3  # Number of clusters
km_model <- kmeans(data, centers = k)

# Step 3: View the cluster results
print(km_model)

# Step 4: Visualize clusters (if applicable)
plot(data, col = km_model$cluster)
points(km_model$centers, col = 1:k, pch = 8, cex = 2)  # Plot cluster centers
```

Output:

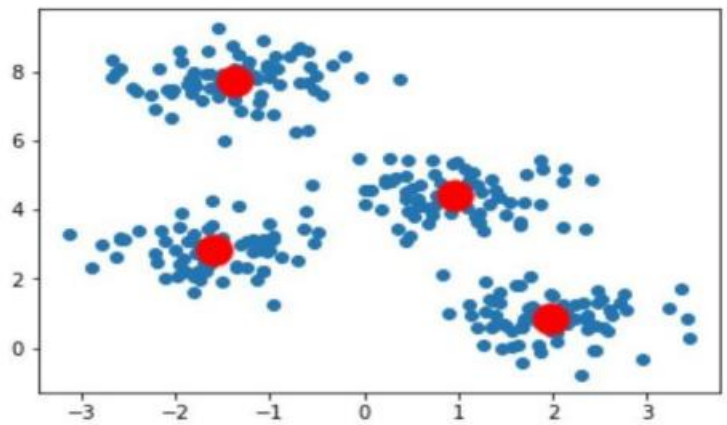
```
K-means clustering with 3 clusters of sizes 17, 15, 18

Cluster means:
        [,1]       [,2]
1  1.0586072  0.2993084
2 -0.2215562 -0.8066471
3 -0.7196001  0.7962155

Clustering vector:
 [1] 3 2 1 3 2 1 2 3 3 3 1 2 2 2 2 1 1 3 1 3 2 2 3 2 2 3 1 2 3 1 1 3 1 1 1 1 1 3
[39] 2 3 3 3 3 1 1 2 3 3 1 2

Within cluster sum of squares by cluster:
[1] 10.389554  8.452049 12.663179
 (between_SS / total_SS =  61.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Experiment 14
Write a program of cluster analysis using simple k-means algorithm Python/R programming language.

```
# K-Means Clustering in R

# Step 1: Load dataset
data(iris)
head(iris)

# Step 2: Use only numeric columns (exclude species column)
iris_data <- iris[, 1:4]

# Step 3: Scale the data (optional but recommended)
scaled_data <- scale(iris_data)

# Step 4: Apply k-means clustering
set.seed(123)  # For same result every time
k <- 3  # Number of clusters
km_result <- kmeans(scaled_data, centers = k, nstart = 25)

# Step 5: View the result
print(km_result)

# Step 6: Add cluster result to the data
iris$Cluster <- as.factor(km_result$cluster)

# Step 7: Plot the clusters (using first two features)
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Cluster,
    main = "K-Means Clustering", xlab = "Sepal Length", ylab = "Sepal Width", pch = 20)
legend("topright", legend = levels(iris$Cluster), col = 1:3, pch = 20)
```

Output:

```
Output

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
K-means clustering with 3 clusters of sizes 50, 53, 47

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1  -1.01119138  0.85041372   -1.3006301  -1.2507035
2  -0.05005221 -0.88042696    0.3465767   0.2805873
3   1.13217737  0.08812645    0.9928284   1.0141287

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 2 2 3 2 2 2
 [75] 2 3 3 3 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 2 3 3 3 3
[112] 3 3 2 2 3 3 3 3 2 3 2 3 2 3 3 2 3 3 3 3 3 3 2 2 3 3 3 2 3 3 3 2 3 3 3 2 3
[149] 3 2

Within cluster sum of squares by cluster:
[1] 47.35062 44.08754 47.45019
 (between_SS / total_SS =  76.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```



Experiment 15

Write a program to compute/display dissimilarity matrix (for your own dataset containing at least four instances with two attributes) using Python

```
import numpy as np
import pandas as pd
from scipy.spatial.distance import pdist, squareform

# Step 1: Define your own dataset with 4 instances and 2 attributes
# Example data: [Height (in cm), Weight (in kg)]
data = np.array([
   [160, 55],  # Instance 1
   [165, 60],  # Instance 2
   [170, 65],  # Instance 3
   [175, 70]   # Instance 4
])

# Step 2: Create DataFrame for better display
df = pd.DataFrame(data, columns=["Height", "Weight"],
                  index=["Person 1", "Person 2", "Person 3", "Person 4"])

print("Dataset:")
print(df)

# Step 3: Compute the dissimilarity (Euclidean distance) matrix
distance_matrix = squareform(pdist(data, metric='euclidean'))

# Step 4: Convert matrix to DataFrame for display
dissimilarity_df = pd.DataFrame(distance_matrix,
                                columns=df.index,
                                index=df.index)

print("\nDissimilarity Matrix (Euclidean distance):")
print(dissimilarity_df.round(2))
```

Output:

```
Dataset:
          Height  Weight
Person 1     160      55
Person 2     165      60
Person 3     170      65
Person 4     175      70

Dissimilarity Matrix (Euclidean distance):
          Person 1  Person 2  Person 3  Person 4
Person 1      0.00      7.07     14.14     21.21
Person 2      7.07      0.00      7.07     14.14
Person 3     14.14      7.07      0.00      7.07
Person 4     21.21     14.14      7.07      0.00
```

Experiment 16
Visualize the datasets using matplotlib in python/R.(Histogram, Box plot, Bar chart, Pie chart etc.,)

Code for pie chart:

```
import matplotlib.pyplot as plt
# initializing the data
x = [10, 20, 30, 40]
y = [20, 25, 35, 55]
# plotting the data
plt.plot(x, y)
plt.show()
```

Output: